

# Unsupervised Geometric Learning of Hyperspectral Images

James M. Murphy, Mauro Maggioni

---

## Abstract

The problem of unsupervised learning and segmentation of hyperspectral images is a significant challenge in remote sensing. The high dimensionality of hyperspectral data, presence of substantial noise, and overlap of classes all contribute to the difficulty of automatically segmenting and clustering hyperspectral images. In this article, we propose an unsupervised learning technique that combines a density-based estimation of class modes with partial least squares regression (PLSR) on the learned modes. The density-based learning incorporates the geometry of the hyperspectral data by using diffusion distance to promote learning a unique mode from each class. These class modes are then used to generate class cores which approximate training labels. Partial least squares regression using these learned class cores as labeled training points consequently determines a labeling of the entire dataset. The proposed method is shown to perform competitively against state-of-the-art clustering and dimension reduction methods, and often achieves performance comparable to fully supervised PLSR.

---

## 1. Introduction

### *1.1. Hyperspectral learning and previous work*

Hyperspectral imagery has emerged as a significant data source in a variety of scientific fields, including medical imaging [1], chemical analysis [2], and remote sensing [3]. Compared to other sensors, hyperspectral sensors capture reflectance at a sequence of localized electromagnetic ranges, allowing for precise differentiation of materials according to their spectra. Indeed, the power of hyperspectral imagery (HSI) for material discrimination has led to their proliferation, making manual analysis of hyperspectral data infeasible in many cases. The large data size of HSI, combined with their high dimensionality, demands innovative methods for storage and analysis. In particular,

efficient machine learning algorithms are needed to automatically process and glean insight from the deluge of hyperspectral data now available.

A variety of hyperspectral learning problems have been investigated with gusto in recent years. The problem of *HSI classification* asks for an algorithm to label each pixel in a given HSI as belonging to a particular class, where training samples from each class are provided together with their labels, to help the algorithm build an effective discrimination tool. A variety of machine-learning techniques have been used for HSI classification, including nearest-neighbor and nearest subspace methods [4, 5], support vector machines [6, 7], neural networks [8, 9, 10] and regression methods [11, 12]. The problem of *HSI unmixing* [13] assumes that the pixels in an HSI scene are constituted from a set of materials, sometimes called endmembers. It is then of interest to determine quantities related to those materials, such as number of endmembers, their spectral signatures, and their abundances in a given pixel. Techniques in matrix factorization [14, 15] and sparse regression [16, 17] have been applied successfully to the problem of hyperspectral unmixing.

The problem of hyperspectral clustering is similar to the problem of hyperspectral classification, except that no training data is available. Hence, decisions about class labels must be made without direct access to elements of the classes themselves. This increases the challenge of the problem considerably, and also introduces the problem of how success ought to be measured; several methods of quality evaluation will be considered in Section 3. Recent techniques for hyperspectral clustering include those based on hierarchical methods [18], particle swarm optimization [19], density analysis [20], Gaussian mixture models (GMM) [21], nearest neighbor clustering [22], and fast search and find of density peaks clustering (FSFDPC) [23, 24, 20]. The proposed method combines density-based methods with geometric learning through diffusion geometry [25, 26] in order to analyze regions of the data that, with high probability, have a common class label. This information is then used for PLSR to compute a low-dimensional embedding that discriminates between the classes, to achieve an accurate and interpretable labeling.

### 1.2. Overview of proposed method

The proposed method is provided with the data  $X = \{x_n\}_{n=1}^N$  and the number  $K$  of classes, and proceeds in two steps:

1. **Mode Identification:** This unsupervised step consists first in performing density estimation and finding  $K$  *modes*  $\{x_i^*\}_{i=1}^K$ , one for each

class. Then, for each  $i$ , a set of points  $C_i^*$  that we label as being in class  $i$  is generated.

2. **Learning a Classifier:** In this supervised step, we use a classification or regression technique (e.g. PLSR) to learn a small set of good predictors for the labels  $1, \dots, K$  from the training set of labeled sets  $\{(C_i^*, i)\}_{i=1}^K$ , and a classifier that depends on those predictors.

By a mode, we mean a point of high density within a class, that we assume is representative of the entire class. We assume  $K$  is known, but otherwise we have no access to labeled data; in Section 5, we will discuss a method for learning  $K$  automatically. The procedure is overall unsupervised, but employs a first unsupervised step to produce putative labels for a small subset of the classes, and then uses a supervised technique to label all other points. The motivation for this approach is to attain robustness with respect to the shape of the distributions corresponding to the different classes, as well as to high-dimensional noise. The modes, suitably defined via density estimation, are robust to noise, and the process we use to pick only one mode per class is based on diffusion distances, which are robust to the shape of the support of the density of a class. The second, supervised, step involves dimension reduction followed by linear regression, which further diminishes the noise in the crucial learning, and then prediction, phases.

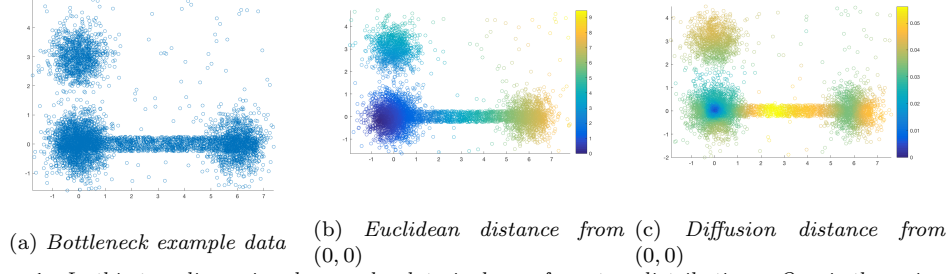
#### 1.2.1. Mode identification

The computation of the modes is a significant part of the proposed method, which we now summarize for a general dataset  $X$ , consisting of  $K$  classes. We make the assumption that modes of the constituent classes can be characterized as a set of points  $\{x_i^*\}_{i=1}^K$  such that

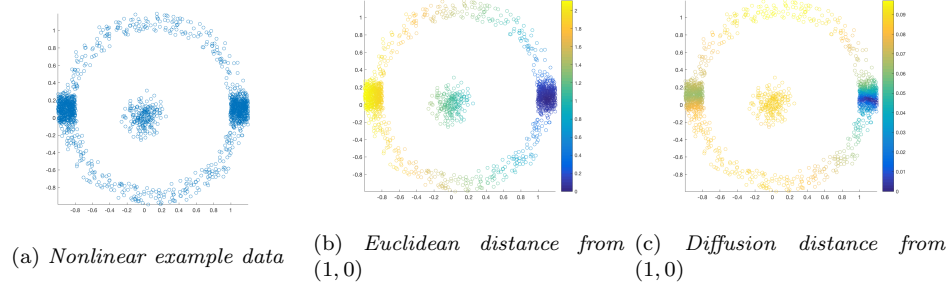
1. the empirical density of each  $x_i^*$  is relatively high;
2. the diffusion distance between pairs  $x_i^*, x_{i'}^*$ , for  $i \neq i'$ , is relatively large.

The first assumption is motivated by the fact that points of high density ought to have nearest neighbors corresponding to a single class; the modes should thus produce neighborhoods of points that with high confidence belong to a single class. However, there is no guarantee that the  $K$  densest points will correspond to unique classes in the case when some classes have a multi-modal distribution. The second assumption addresses this issue, and is motivated by the fact that points may be far away in Euclidean distance,

even if they are drawn from the same distribution, whereas they may be closer in diffusion distance. Figures 1 and 2 illustrate this phenomenon.



**Figure 1:** In this two-dimensional example, data is drawn from two distributions. One is the union two isotropic Gaussians with means at  $(0,0)$  and  $(6,0)$ , respectively, connected by a set of points uniformly sampled from a thin rectangle. This bridge forms a bottleneck in the distribution. The second distribution is an isotropic Gaussian with mean at  $(3,0)$ . We plot the distances from the point  $(0,0)$  in the Euclidean and diffusion distances, respectively. The rectangle acts as a bridge between the two Gaussians and causes the high density points to be close in diffusion distance, compared to their distance in the usual  $\ell^2$  i.e. Euclidean norm. Indeed, the bottleneck is overcome efficiently with diffusion distance.



**Figure 2:** In this two-dimensional example, data is drawn from two distributions. One is an isotropic Gaussian with mean  $(0,0)$ , the other is supported on a thin annulus around  $(0,0)$ , with two high-density regions around  $(-1,0)$  and  $(1,0)$  respectively. We plot the distances from the point  $(1,0)$  in Euclidean and diffusion distances, respectively. With diffusion distances, the two high density regions on the annulus are closer to each other than either is to the center high density region. This is due to the large number of paths with small step sizes connecting the high density regions on the annulus.

Mathematically, one can understand  $X = \{x_n\}_{n=1}^N$  as sampled from some distribution.

$$\mu = \sum_{i=1}^K w_i \mu_i,$$

where each  $\mu_i$  corresponds to the probability distribution of class  $i$ , and the weights  $\{w_i\}_{i=1}^K$  correspond to how often each class is sampled, and naturally  $\sum_i w_i = 1$ . More precisely, sampling  $x \sim \mu$  means first sampling  $Z \sim \text{Multinomial}(w_1, \dots, w_K)$ , then sampling from  $\mu_m$ , conditioned on the event  $Z = m$ .

The mode identification algorithm produces a point  $x_i^*$  corresponding to each  $\mu_i$ . Enforcing that these modes are far apart in diffusion distance has several advantages over Euclidean distance. The proposed method has the ability to acquire a single mode from each class, even in the case that certain classes are multimodal with several modes at higher density than the densest point in another class. Diffusion distances are also independent of shape of distribution, and are thus suitable for identifying nonlinear clusters. These advantages of diffusion distance for mode detection are illustrated with a toy example in Figure 3.

At this point we assume that we have found exactly one mode  $x_i^*$  for each class. Let  $C_i^*$  be a set of nearest neighbors to  $x_i^*$  as measured, for example, in the  $\ell^2$  norm or diffusion distance, noting that for small distances the two are nearly equivalent. In an ideal setting, each  $C_i^*$  contains points from only a single class, and each class is represented among the  $\{C_i^*\}_{i=1}^K$  exactly once. These points may in this case be understood as “learned training points,” which may then be deployed in a supervised regression method. The entire process for learning these points was unsupervised, but results in a set of points that can be used in a supervised algorithm. Note that these points are quite different from the types of points typically used in a supervised classification method. Indeed, training points in the supervised setting are generated by randomly sampling a set of known labels. Hence, these points ought to be well-distributed within the class, with respect to the underlying class distribution. The points generated by our method, in distinction, are the nearest neighbors of a relatively high density point, and hence are unlikely to be well-distributed within the class. Rather than a representative sample of the entire class, the learned modes may be understood as points that, with high confidence, belong to the same class. Examples comparing the learned training labels to randomly selected ground truth labels appear in Figures 9, 12, 15, 18, and 21.

### 1.2.2. Learning a classifier

Once the  $\{C_i^*\}_{i=1}^K$  have been learned, they are used as labels in the construction of a supervised estimator for the class labels. In this work we consider a vector-valued version of the well-known partial least squares regression (PLSR) [27, 28, 29, 30]. PLSR is a linear regression method that transforms the predictor and response variables in such a way that their variation is jointly maximized. This is in contrast with least squares regression (LSR) [30, 31] in which no transformation is applied to the predictor vari-

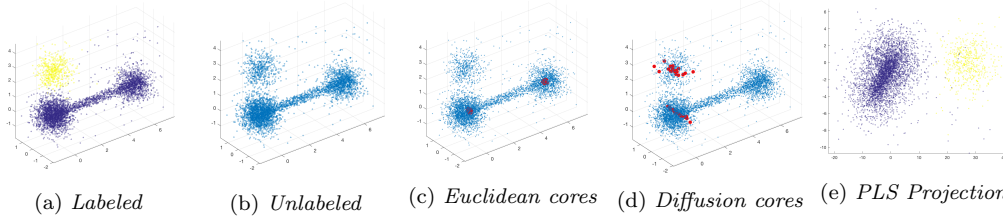
ables, and principal component regression (PCR) [30], in which the transformation maximizes only the variance in the predictor variables. For learning problems in which the ultimate goal is labeling points, transforming both the predictor and response variables to maximize both variance and discrimination between classes is more convenient than maximizing variance alone. PLSR has been seen to be useful for the analysis of HSI movies [2], among other applications to the analysis of HSI [32, 33], and is one of the most used techniques in chemometrics [30, 34].

## 2. Algorithm

### 2.1. Motivating example and approach

A key aspect of our algorithm is our method for identifying the modes of the underlying classes in the HSI data. This is a significant challenge, due to the high ambient dimensionality of HSI, potential overlap between distributions at their tails, along with differing densities, sampling rates, and distribution shapes across classes. Consider the simplified example in Figure 3. This data set consists of a point cloud in  $\mathbb{R}^3$  generated as a realization of a mixture model of two distributions; the first is unimodal, consisting of a single Gaussian, while the second is multimodal, consisting of two Gaussians connected by a bridge of points. There is also a small amount of uniform noise added.

The points of high density lie close to the center of the first distribution, and close to one of the two ends of the second distribution. After computing an empirical density estimate, the distance between high density points is computed. If the Euclidean distance between high density points is used to remove spurious modes, i.e. modes corresponding to the same distribution, then the two modes on opposite sides of the second distribution are returned, which incorrectly assigns two modes to the same distribution. However, if diffusion distance is used, rather than Euclidean distance, then the two modes learned correspond to two different classes. This is because the modes on the opposite ends of the second distribution are far in Euclidean distance but closer in diffusion distance. Indeed, the points between the two modes act as a bridge, which is incorporated into the geometry learned by diffusion distances. On the other hand, there is a more substantial region of low density between the two distributions, making the diffusion distance between them relatively large. This suggests that diffusion distance is more useful than Euclidean distance for comparing high density points for the determination



**Figure 3:** In the following three-dimensional example, data is drawn from two distributions: the first is unimodal, while the second is bimodal. The goal is to accurately learn two modes from the mixture in a totally unsupervised manner, and each mode ought to correspond to a single distribution. We consider two variations of mode estimation. First, we compute distance between high density points with Euclidean distance, which yields an incorrect result, computing two modes from the same distribution. Next, we compute distance between high density points with diffusion distance, which correctly captures one mode from each distribution. Despite the fact that the diffusion learned modes are close in Euclidean distance, they are far in diffusion distance, and more suitable for distribution core estimation. Notice that the distributions are very well-separated once projected onto the learned PLS directions, excepting a few noise points.

of modes, under the assumptions that multimodal regions have modes that are connected by regions of not too low density. From the standpoint of Euclidean distance, the bridge is insignificant. From the standpoint of diffusion distance, it is critical.

## 2.2. Algorithm description

We now discuss the proposed algorithm in detail. Let  $X$  be the HSI image, reshaped into an  $N \times D$  matrix, where  $N$  is the number of pixels and  $D$  is the number of spectral bands. We may consider  $X$  as a collection of points  $\{x_n\}_{n=1}^N \subset \mathbb{R}^D$ . As described in Section 1.2, our algorithm proceeds in two major steps: mode identification and learning of a classifier.

The algorithm for learning the modes of the classes first computes an empirical density for each point with a kernel density estimator. This computes, for each  $n \in \{1, \dots, N\}$ ,

$$p(x_n) = \sum_{x_m \in NN(x_n)} e^{\frac{-\|x_n - x_m\|_2^2}{\sigma^2}},$$

where  $\|x_n - x_m\|_2^2$  is the squared  $\ell^2$  norm in  $D$ -dimensions.

The set  $NN(x_n)$  is the set of  $k$ -nearest neighbors to  $x_n$ , in the  $\ell^2$  norm. In Section 3,  $k = 20$ , though our method is robust to choosing larger  $k$ . The parameter  $\sigma$  in the exponential kernel is data-adaptive, and set to be half the mean distance between all points, which makes this choice relatively robust across different data sets. The density  $p$  is normalized so that  $\sum_{n=1}^N p(x_n) = 1$ .

Once this empirical density  $p$  is computed, the modes of the HSI classes are computed in a manner similar in spirit to [23]. We compute the quantity  $\rho$  that assigns, to each pixel, the minimum diffusion distance between the pixel and a point of higher empirical density. The point with highest empirical density is assigned the maximum distance between it and any other point as its  $\rho$  value. More precisely, we compute

$$\rho(x_n) = \begin{cases} \min\{d_t(x_n, x_m) \mid p(x_m) \geq p(x_n)\}, & x_n \neq \arg \max_i p(x_i), \\ \max_j d_t(x_n, x_j), & x_n = \arg \max_i p(x_i). \end{cases}$$

where  $d_t(x_m, x_n)$  is the diffusion distance between  $x_m, x_n$ , at time  $t$ . Details of the diffusion distance construction are given in Section 6. The quantity  $\rho$  is normalized to have maximum value 1. For comparative purposes, we also consider computing  $\rho$  with the usual  $\ell^2$  distance.

The modes of the HSI are computed as the points  $x_1^*, \dots, x_K^*$  yielding the  $K$  largest values of the quantity

$$\mathcal{D}(x_n) = p(x_n)\rho(x_n)$$

Intuitively, such points should be both high density and far in diffusion distance from any other higher density points, and can therefore be expected to be modes of different distributions. From the learned modes  $\{x_i^*\}_{i=1}^K$ , approximate cores of distributions  $\{C_i^*\}_{i=1}^K$  are learned as the  $k$ -nearest neighbors in the  $\ell^2$  or diffusion norm of each  $x_i^*$ . We set  $k$  data-adaptively as  $\lfloor .02 \cdot N \rfloor$ ; this allows for the size of the cores to scale linearly with  $N$ . Our method is robust to small changes in core size, however. We summarize the core detection algorithm in Algorithm 1.

**Algorithm 1:** Core Detection Algorithm

- 1 Compute the empirical densities  $\{p(x_n)\}_{n=1}^N$  for all elements of  $X$ .
- 2 Compute  $\{\rho(x_n)\}_{n=1}^N$ , the diffusion (Euclidean) distance from each point its nearest neighbor in diffusion (Euclidean) distance of higher empirical density.
- 3 Set the learned modes  $\{x_i^*\}_{i=1}^K$  to be the  $K$  maximizers of  $\mathcal{D}(x_n) = p(x_n)\rho(x_n)$ .
- 4 Set the  $i^{th}$  learned core  $C_i^*$  to be the  $k$  nearest neighbors of  $x_i^*$  as measured in diffusion (Euclidean) distance.



The cores  $\{C_i^*\}_{i=1}^K$  are then used as labels for PLSR. Instead of arbitrarily assigning numbers to each of the classes and regress these, we assign the elements of  $C_i^*$  to the vertex of a  $K$ -simplex, in order to perform vector-valued regression with PLSR without imposing a one-dimensional structure to the space of labels. More precisely, the points in  $C_i^*$  are labeled as  $\mathbb{1}_i = (0, 0, \dots, 0, 1, 0, \dots, 0)$  with 1 in the  $i^{th}$  coordinate. This generates a set of training pairs for the  $i^{th}$  class:

$$T_i = \{(x, y) \mid x \in C_i^*, y = \mathbb{1}_i\}.$$

This training data  $T = \bigcup_{i=1}^K T_i$  is used in PLSR to acquire a regression coefficient  $\hat{\beta}$ ; details for the PLSR construction appear in Section 6.2. The regression coefficient  $\hat{\beta}$  is a  $D \times K$  matrix that is derived only from the learned cores. Multivariate regression on all of  $X$  is performed by computing  $\hat{Y} = \hat{\beta}X$ ;  $\hat{Y}$  is an  $N \times K$  matrix with rows corresponding to points and columns corresponding to responses of each point. The columns may be interpreted as affinity for each class. Class labels are learned as

$$c(n) = \arg \max_{i=1, \dots, K} \hat{Y}_{n,i}.$$

Other methods for computing class labels from the responses  $\hat{Y}$  were considered, including  $k$ -means on  $\hat{Y}$  with  $k = K$ , yielding similar results. The overall algorithm is summarized in Algorithm 2.

<b>Algorithm 2:</b> Unsupervised Clustering Algorithm
<ol style="list-style-type: none"> <li>1 Learn cores of <math>X</math> according to Algorithm 1, call them <math>\{C_i^*\}_{i=1}^K</math>.</li> <li>2 Learn PLSR coefficient <math>\hat{\beta}</math> using the learned cores as training data.</li> <li>3 Regress <math>X</math> as <math>\hat{Y} = \hat{\beta}X</math>.</li> <li>4 Assign labels to all data points as <math>c(n) = \arg \max_{i=1, \dots, K} \hat{Y}_{n,i}</math>.</li> </ol>



### 3. Experiments

#### 3.1. Algorithm Evaluation Methods and Experimental Data

We consider several HSI datasets to evaluate the proposed algorithm. For evaluation, we consider three quantitative measures, as well as visual performance. The quantitative measures of performance are:

1. *Overall accuracy (OA)*: Total number of correctly labeled pixels divided by the total number of pixels. This method values large classes more than small classes.
2. *Average accuracy (AA)*: For each class, the overall accuracy of that class is computed, as described above. These class-wise performances are then averaged. This method values small classes and large classes equally.
3. *Cohen’s  $\kappa$ -statistic*: A measurement of interrater agreement that measures agreement between two labelings, corrected for random agreement [35]. Letting  $a_o$  be the observed agreement between the labeling and the ground truth and  $a_e$  the expected agreement between a uniformly random labeling and the ground truth,  $\kappa = \frac{a_o - a_e}{1 - a_e}$ . A  $\kappa$  value of 1 corresponds to perfect overall accuracy, while a labeling that does not improve on what is expected from random guessing would have  $\kappa \leq 0$ .

In order to perform quantitative analysis with these metrics, ground truth is required, and the learned clusters must be aligned with it. After performing our unsupervised clustering, we perform an a posteriori alignment of the labels with the ground truth that maximizes the overall accuracy of the labeling. More precisely, let  $S_K$  be the symmetric group on  $K$  letters, that is, the group of permutations of  $\{1, 2, \dots, K\}$ . Let  $\{C_i\}_{i=1}^K$  be the clusters learned from one of the clustering methods, and let  $\{C_i^{GT}\}_{i=1}^K$  be the ground truth clusters. For error evaluation and visualization, cluster  $C_i$  is assigned label  $\hat{\eta}_i \in \{1, 2, \dots, K\}$ , where  $\hat{\eta} = (\hat{\eta}_1, \dots, \hat{\eta}_K)$  is determined as

$$\hat{\eta} = \arg \max_{\eta = (\eta_1, \dots, \eta_K) \in S_K} \sum_{i=1}^K |C_{\eta_i} \cap C_i^{GT}|.$$

We remark that this alignment method maximizes the overall accuracy of the labeling and is most useful for visualization, but more optimal labelings for maximizing  $AA$  and  $\kappa$  could exist.

First, we consider synthetic data, where classes are generated randomly according to a specified distribution. Next, we consider six real HSI datasets which reflect strengths and weaknesses of the proposed algorithm. Five of the datasets are standard and have ground truth; they are publicly available at [http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral\\_Remote\\_Sensing\\_Scenes](http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes). We also consider a dataset without ground truth, for

which the proposed quantitative measures do not apply. We summarize the datasets, along with whether ground truth is available, as:

1. Synthetic (GT)
2. Kennedy Space Center (GT)
3. Pavia (GT)
4. Indian Pines (GT)
5. Pavia University (GT)
6. Salinas A (GT)
7. Chemical Plume (no GT)

In order to reduce the number of classes to be suitable for our unsupervised method, the images are restricted to subsets in the spatial domain. Moreover, for the data with ground truth, not all pixels are labeled in the ground truth. Experiments are only performed on the labeled portion of the data, in order to perform meaningful quantitative analysis and to be able to use the supplied number of classes  $K$  as a parameter. Images of ground truth and the sum of all bands for each dataset are in Figures 7, 10, 13, 16, and 19. The sum of all bands is presented as a simple summary of the data. Finally, we consider an HSI image containing a chemical plume. This image does not have ground truth, so the goal is to cluster it so that the chemical plume is clearly segmented.

All experiments and subsequent analyses are done in MATLAB; code to reproduce all results will be made publicly available on <http://www.math.jhu.edu/~jmurphy/>.

### 3.2. Comparison Methods

We consider several methods of analysis for comparison. First, we consider the classic *K-means* algorithm [30] applied directly to  $X$ . This method is not expected to generate especially good results for HSI data, in part due to the non-spherical shape of clusters in HSI data. Non-spherical classes is a well-known problem for *K-means* clustering. We also consider a variant of *K-means* in which the initial centroid guesses are not random, but the learned modes from the proposed mode detection algorithm, as we expect that these initial guesses are more suitable than random initializations. Of course, multiple runs of *K-means* are expected to output results close to the optimum, at least when the clusters have shapes that are amenable of being well-separated by *K-means*.

One aspect of HSI data that confounds straightforward clustering methods is the high ambient dimensionality of the dataset, as well as the presence of non-negligible noise. Several methods to reduce the dimensionality of the data, while preserving important discriminatory properties of the classes, as well as increasing the signal-to-noise ratio in the projected space, are used for comparison. We first consider dimension reduction via *principal component analysis*, which seeks to find a low-dimensional representation of the dataset that retains maximum variance. We then consider *independent component projection* [36, 37], which seeks to decompose the data into maximally independent components. In addition, we consider dimensionality reduction by a *random projection* via a Gaussian random matrix, which has been shown to be efficient in highly simplified data models [38, 39]. All of these methods involve first reducing the dimension,  $D$ , of the data, then running  $K$ -means on the reduced data. The datasets are reduced to being  $K_{GT} \ll D$  dimensional, where  $K_{GT}$  is the number of classes, if known. In the case of chemical plume detection, when the number of classes is unknown,  $K_{GT}$  is estimated; see Section 3.9 for details.

We also consider *spectral clustering* [40, 41], which has had success in classifying and clustering HSI [42]. The spectral embedding was computed using the normalized graph Laplacian with 100 nearest neighbors and weight function  $W_{i,j} = e^{-\|x_i - x_j\|_2^2}$ . We also consider clustering with *Gaussian mixture models* (GMM) [43, 44, 21]. This method attempts to fit a number of Gaussians to the dataset, with means and covariance matrices determined by expectation maximization (EM). Moreover, we consider the *fast search and find of density peaks clustering* (FSFDPC) algorithm [23], which has proven effective in clustering certain data. The original implementation, with Euclidean distances to compare the distances between high density points, is used as a comparison method. We also propose a novel modification using diffusion distances. It is hypothesized that this variation of FSFDPC will outperform the version with Euclidean distances.

Finally, we consider PLSR with cores learned both with Euclidean distances and diffusion distances. As in the case of FSFDPC, it is hypothesized that diffusion distance will be more effective than Euclidean distance at discriminating between high density points, and will thus produce more accurate clustering results. In total, we consider 12 unsupervised methods, along with 1 supervised method, which are summarized in Table 1. To account for the randomness of training data generation, and also the randomness of certain unsupervised comparison techniques, each experiment was repeated 10 times

and numerical results were averaged. The displayed visual results are from a single trial, and are generally representative of the particular experiment, since there is little variance in even the random methods across different trials.

The FSFDPC method has similarities with the mode estimation aspect of the present work, in that both algorithms attempt to learn the modes of the data classes via a density based analysis, as described in, for example, [23, 20]. Our method is quite different from this approach, however. In the density analysis, we use a kernel density estimator, rather than a hard-cut off, which improves robustness and reduces the need for data-dependent tuning. More significantly, the proposed measure of distance between high density points is not Euclidean distance, but diffusion distance [25, 26], which is more adept at removing spurious modes, due to its incorporation of the geometry of the data. These labels are then used as training data for PLSR, which will learn a linear embedding of the data that optimally discriminates both the predictor and response variables associated to these points. The FSFDPC algorithm, in contrast, simply assigns each of the modes its own class label, then assigns the remaining points a label by requiring that each point has the same label as its nearest neighbor of higher density. In this regard, FSFDPC algorithm is instance-based, and makes all labeling decisions locally, whereas the proposed method is global. It is thus expected that the proposed algorithm will be less prone to overfitting than FSFDPC, and will be more useful on challenging datasets when classes are not easily separable.

Method	Supervised	Dimension Reduction	Distance Metric	Novel
<i>K</i> -means on full dataset with random initializations	No	No	Euclidean	No
<i>K</i> -means on full dataset with learned Euclidean modes as centroid initializations	No	No	Euclidean	No
<i>K</i> -means on full dataset with learned diffusion modes as centroid initializations	No	No	Diffusion	Yes
<i>K</i> -means on PCA reduced dataset	No	Yes	Euclidean	No
<i>K</i> -means on ICA reduced dataset	No	Yes	Euclidean	No
<i>K</i> -means on data reduced by random Gaussian projection	No	Yes	Euclidean	No
Spectral clustering	No	Yes	Euclidean	No
Gaussian mixture models	No	No	Euclidean	No
FSFDPC with Euclidean distance	No	No	Euclidean	No
FSFDPC with diffusion distance	No	No	Diffusion	Yes
PLSR with Euclidean learned cores	No	Yes	Euclidean	Yes
PLSR with diffusion learned cores	No	Yes	Diffusion	Yes
PLSR with ground truth	Yes	Yes	Euclidean	No

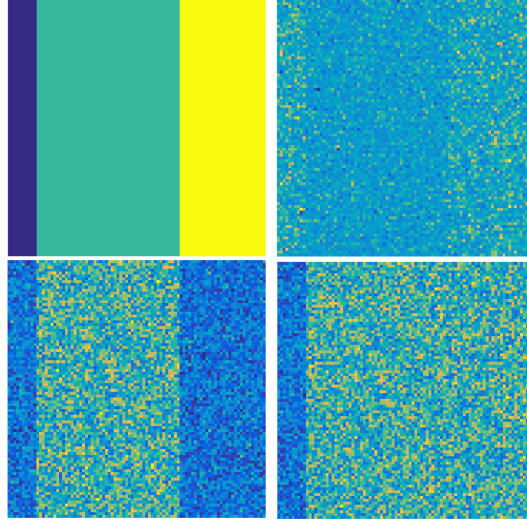
**Table 1:** *Methods used for experimental analysis*

The novel methods proposed are *K*-means with diffusion learned modes as centroid initializations, FSFDPC with diffusion distance, PLSR with Euclidean learned cores, and PLSR with diffusion learned cores. While *K*-means with diffusion learned modes as centroid initializations is novel, the idea of seeding *K*-means intelligently is known, and the proposal of this technique is not considered a significant contribution of the present article. The primary

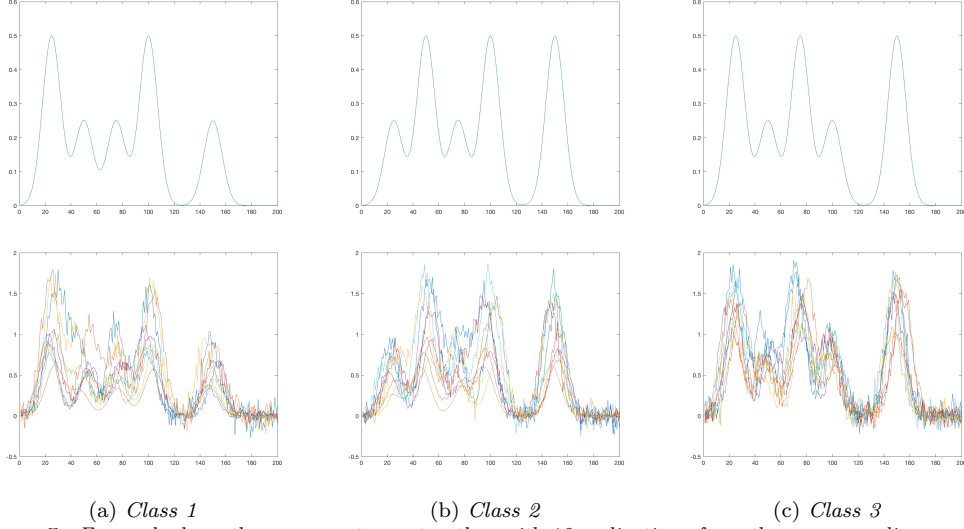
proposed method is PLSR with diffusion cores; the other novel methods are presented to validate the use of diffusion distances and PLSR. It is hypothesized that FSFDPC with diffusion distance will outperform FSFDPC with Euclidean cores in most cases. Moreover, it is hypothesized that PLSR will outperform FSFDPC in more challenging examples, where FSFDPC has a tendency to overfit the data.

### 3.3. Synthetic Dataset

The synthetic HSI dataset is size  $90 \times 90 \times 200$ , with 3 classes generated by drawing samples from a random distribution. Classes constitute spatial rectangles of size  $10 \times 90$ ,  $50 \times 90$ , and  $30 \times 90$  respectively; some sample bands appear in Figure 4. Since the algorithms we discuss do not use spatial regularity or correlations, there is no loss of generality in using rectangular regions for the classes. We fix three pure spectra for these three classes, then randomly perturb the locations and values of their maxima, randomly dilate the spectra, and add random noise. Some typical spectra for each class appear in Figure 5.



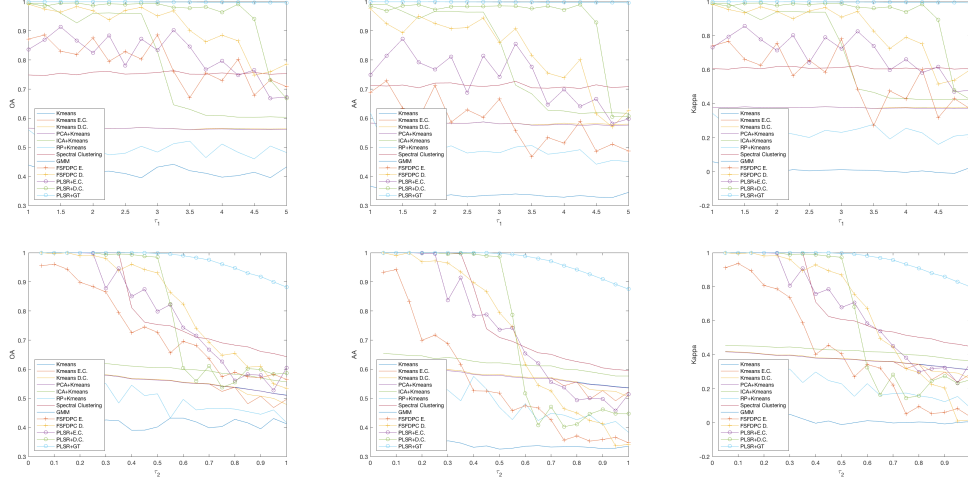
**Figure 4:** *Synthetic HSI data. Clockwise from top left: ground truth, band 10, band 50, band 150. In certain bands, some of the classes are indistinguishable; in others, they are well separated.*



**Figure 5:** For each class, the pure spectrum, together with 10 realizations from the corresponding random distributions, are shown. To generate the random data, the pure spectra have the locations of their maxima perturbed, are scaled by a constant factor, and are corrupted by Gaussian noise. The extent of the perturbations of the extrema are controlled by the  $\tau_1, \tau_2$  parameters. In this case,  $\tau_1 = 4, \tau_2 = .5$ .

More precisely, an element of the  $i^{th}$  class is generated from a pure spectrum  $s_i, i = 1, 2, 3$ , as follows. Each spectrum has five local maxima. For each  $i$ , let  $\{(\gamma_j^i, \alpha_j^i)\}_{j=1}^5$  be the location of the spectra maxima for the pure spectrum  $s_i$ . Let  $\tau_1, \tau_2$  be parameters. A sample from class  $i$  is generated as follows:

1. Randomly translate the  $\{\gamma_j^i\}_{j=1}^5$  independently according to a uniform distribution  $\text{Unif}([- \tau_1, \tau_1])$ , then smooth.
2. Randomly shift the  $\{\alpha_j^i\}_{j=1}^5$  independently according to a uniform distribution  $\text{Unif}([- \tau_2, \tau_2])$ , then smooth.
3. Randomly perturb the entire spectra by a pointwise multiplication by  $\beta \sim \text{Unif}([\frac{1}{2}, \frac{3}{2}])$ .
4. Add mean 0, variance .01 Gaussian random noise componentwise.



**Figure 6:** Synthetic data numerical results. The top row corresponds to fixed  $\tau_2$  and varying  $\tau_1$ , while the second row corresponds to fixed  $\tau_1$  and varying  $\tau_2$ . As the varying parameter  $\tau_i$  increases, the classes become harder to distinguish, reducing accuracy with respect to OA, AA, and  $\kappa$ . PLSR with diffusion cores achieves the best accuracy for the longest duration, before decaying to mediocrity. The methods using diffusion distance outperform the methods using Euclidean distance, and the methods using PLSR outperform those using FSFDPC. The supervised method, PLSR with ground truth, performs the best, and continues to perform well after all other methods decay. This is to be expected, since this method does not need to learn cores or modes of the three distributions: the ground truth is always given. However, PLSR with diffusion cores performs comparably to PLSR with ground truth, until the former breaks and begins to decay rapidly.

Results for  $\tau_1 = 4$  fixed and  $\tau_2$  varying, and  $\tau_2 = .5$  and  $\tau_1$  varying appear in Figure 6; a range of other  $\tau_i$  values were also considered, with similar results. As the varying parameter increases, the clustering problem becomes more challenging. Of the unsupervised methods methods, PLSR with diffusion cores performs well for the largest range of  $\tau_i$ , then quickly breaks once a certain sufficiently large  $\tau_i$  is reached. The observed phase transition is due to the failure to identify modes from distinct classes, caused by increased class overlap. The decay of PLSR with diffusion cores is more severe than other methods; this is due to the fact that it performs by far the best, up until the point it breaks. Once it breaks, the method performs basically as well as the other methods. This leads to a step decay rate, compared to the other methods which do not perform as well at their best.

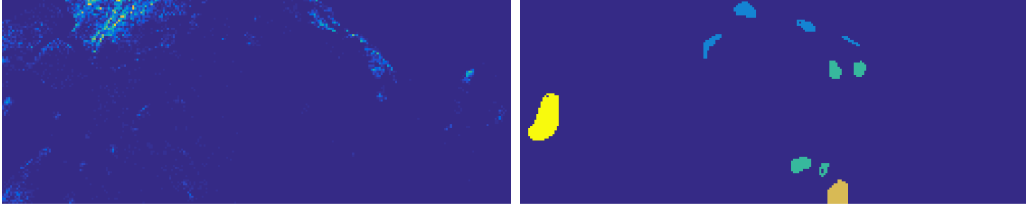
After the  $\tau_i$  parameter is sufficiently large and the classes are very mixed, all algorithms converge to rather poor accuracies. These plots illustrate an important drawback of the proposed method: when the algorithm fails to find distinct modes of the distributions, results may be rather poor. Note that PLSR with Euclidean cores outperforms FSFDPC with Euclidean modes, indicating the value of PLSR over FSFDPC, independent of which method



is used to distinguish between high density points.

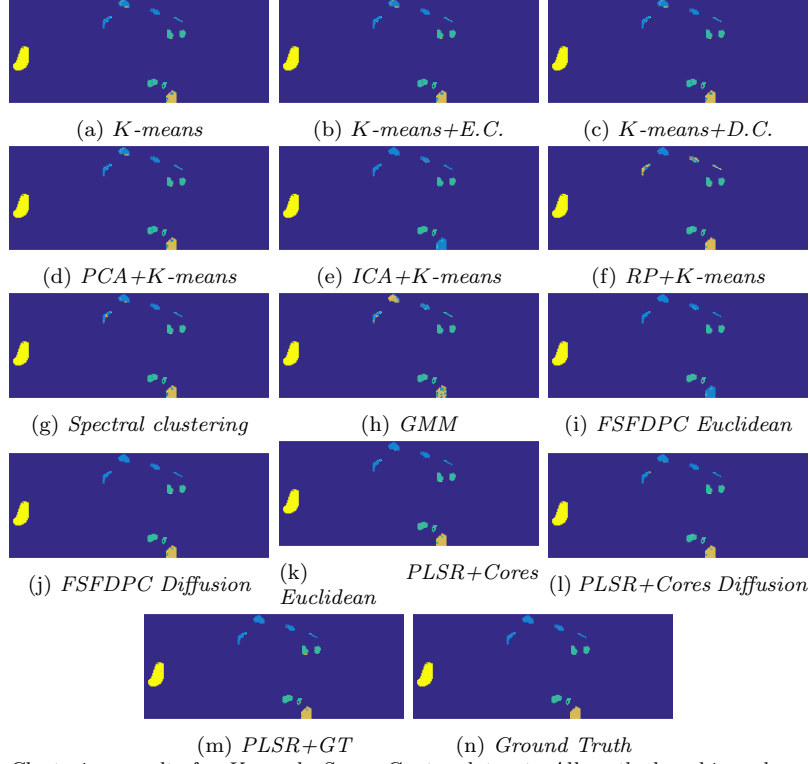
### 3.4. Kennedy Space Center Dataset

Our first real HSI example is the Kennedy space center dataset, which appears in Figure 7. The four classes in this example are quite well separated spatially, so it is hypothesized that all methods will perform well in this example, even the basic  $K$ -means algorithm. Results appear in Figure 8 and Table 2.



**Figure 7:** The KSC data is a  $250 \times 100$  subset of the full KSC dataset. It contains 4 classes that are well localized spatially. The dataset was captured by the NASA AVIRIS sensor over the Kennedy Space Center in FL, USA in March 1996. The spatial resolution is 18m/pixel. There are 224 spectral bands.

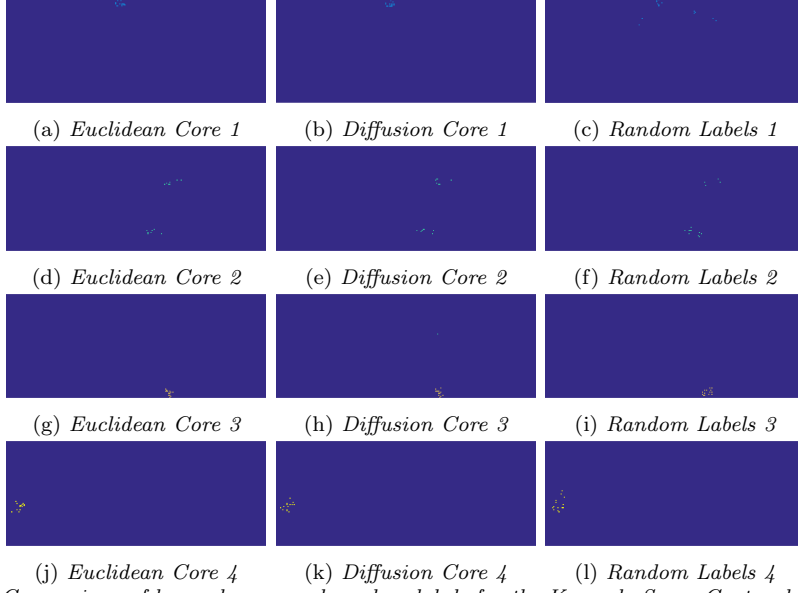
In the Kennedy Space Center (KSC) dataset, all method performed decently, with three methods performing very well: FSFDPC with diffusion cores and both unsupervised PLSR methods. FSFDPC with diffusion cores performed best by a slight margin, and achieved results even exceeding PLSR with GT. The quality of cores learned for this example helps explain this excellent performance: as seen in Figure 9, the learned cores are very similar to the randomly generated training data, with only the first core suffering from some minor spatial localization. The overall strong numerical and visual results suggest that this is a relatively easy dataset to cluster. Notice that the use of diffusion distances over Euclidean distances provides evident improvement in this example.



**Figure 8:** Clustering results for Kennedy Space Center dataset. All methods achieve decent labeling of the clusters. Indeed, all methods correctly label the yellow class. Some mistakes are made on the other classes, but performance appears visually strong across most methods.

Metric	<i>K</i> -means	<i>K</i> -means+E.C.	<i>K</i> -means+D.C.	PCA+ <i>K</i> -means	ICA+ <i>K</i> -means	RP+ <i>K</i> -means	SC	GMM	FSFDPC-E.	FSFDPC-D.	PLSR+E.C.	PLSR+D.C.	PLSR+GT
OA	0.9448	0.9448	0.9448	0.9448	0.8267	0.8899	0.9532	0.8825	0.8390	<b>0.9831</b>	0.9755	0.9770	0.9805
AA	0.9320	0.9320	0.9320	0.9320	0.7377	0.8753	0.9427	0.8662	0.7422	<b>0.9754</b>	0.9684	0.9709	0.9743
$\kappa$	0.9242	0.9242	0.9242	0.9242	0.7584	0.8501	0.9359	0.8408	0.7751	<b>0.9768</b>	0.9663	0.9684	0.9733

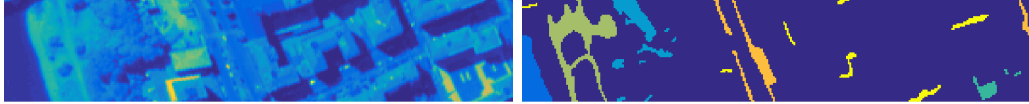
**Table 2:** Quantitative analysis for Kennedy Space Center dataset.



**Figure 9:** Comparison of learned cores and random labels for the Kennedy Space Center dataset. There is little visual difference between the learned cores and the uniformly sampled training data, indicating that core detection was fairly simple in this example.

### 3.5. Pavia Dataset

The Pavia dataset contains six classes, with one of them spread out across the image. As can be seen in Figure 10, the yellow class is small and diffuse, which is expected to add challenge to this example. Moreover, the large number of classes is likely to make it more difficult than the KSC example. Results appear in Figure 11 and Table 3.

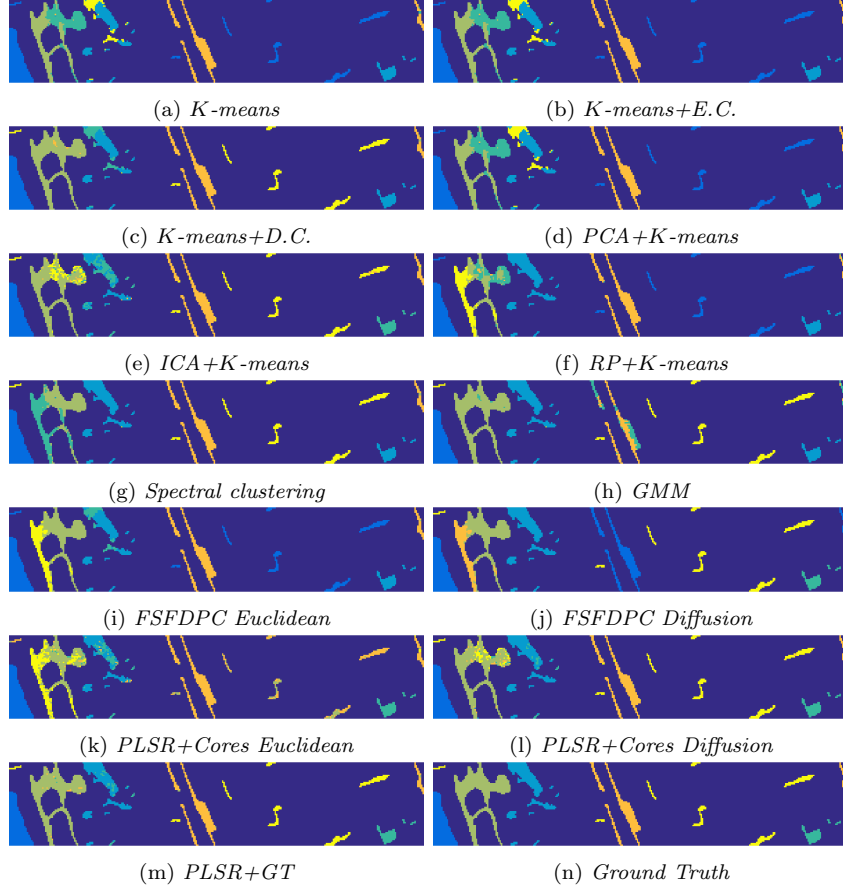


**Figure 10:** The Pavia data is a  $270 \times 50$  subset of the full Pavia dataset. It contains 6 classes, some of which are not well-localized spatially. The dataset was captured by the ROSIS sensor during a flight over Pavia, Italy. The spatial resolution is 1.3 m/pixel. There are 102 spectral bands.

For the Pavia dataset, the PLSR with diffusion cores method performs best, followed by  $K$ -means with diffusion modes. However, the  $K$ -means with diffusion modes algorithm totally mislabels class 3, leading to a substantially lower AA value than PLSR with diffusion cores; the OA scores are comparable between the two methods, since class 3 is relatively small.

As seen in Figure 12, the cores are learned correctly by the PLSR with diffusion cores algorithm, but core 4 is very localized, leading to some labeling errors far away from the core of this class. This causes the PLSR with diffusion cores algorithm to underperform with respect to supervised PLSR

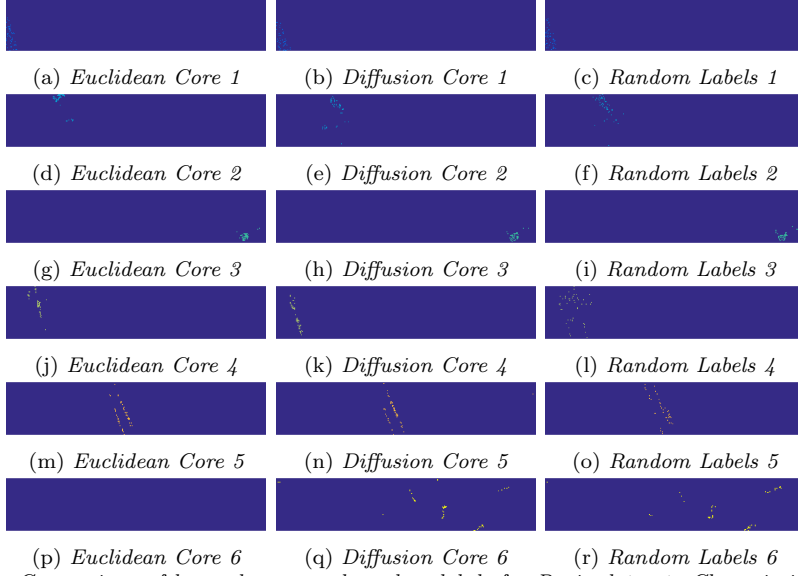
with GT. This indicates a weakness of the proposed method: if a class is large and diffuse, it can be difficult to learn a core for this class that has sufficient within-class variance to be a serviceable substitute for uniformly sampled training data from that class.



**Figure 11:** Clustering result for Pavia dataset. PLSR with diffusion cores performs best, making only small mistakes on two of the classes. Kmeans with diffusion modes performs second best, but crucially completely mislabels the blue-green class on the bottom right of the ground truth. It also makes some mistakes on the blue class in the center right of the ground truth image.

Metric	<i>K</i> -means	<i>K</i> -means+E.C.	<i>K</i> -means+D.C.	PCA+ <i>K</i> -means	ICA+ <i>K</i> -means	RP+ <i>K</i> -means	SC	GMM	FSFDPC-E.	FSFDPC-D.	PLSR+E.C.	PLSR+D.C.	PLSR+GT
OA	0.6502	0.6502	0.8896	0.6502	0.8437	0.6434	0.7942	0.7118	0.7010	0.7169	0.6957	<b>0.9256</b>	0.9592
AA	0.5356	0.5356	0.7731	0.5356	0.8754	0.5444	0.7531	0.6175	0.5688	0.7776	0.7052	<b>0.9504</b>	0.9659
$\kappa$	0.5715	0.5725	0.8597	0.5717	0.8072	0.5642	0.7503	0.6435	0.6279	0.6471	0.6259	<b>0.9068</b>	0.9483

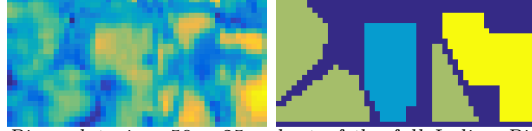
**Table 3:** Quantitative analysis for Pavia dataset.



**Figure 12:** Comparison of learned cores and random labels for Pavia dataset. Class six is not learned with Euclidean distances, but all classes are learned with diffusion distances. Some of the learned diffusion cores are more concentrated than the corresponding uniform sampled training data, for example cores 4 and 5.

### 3.6. Indian Pine Dataset

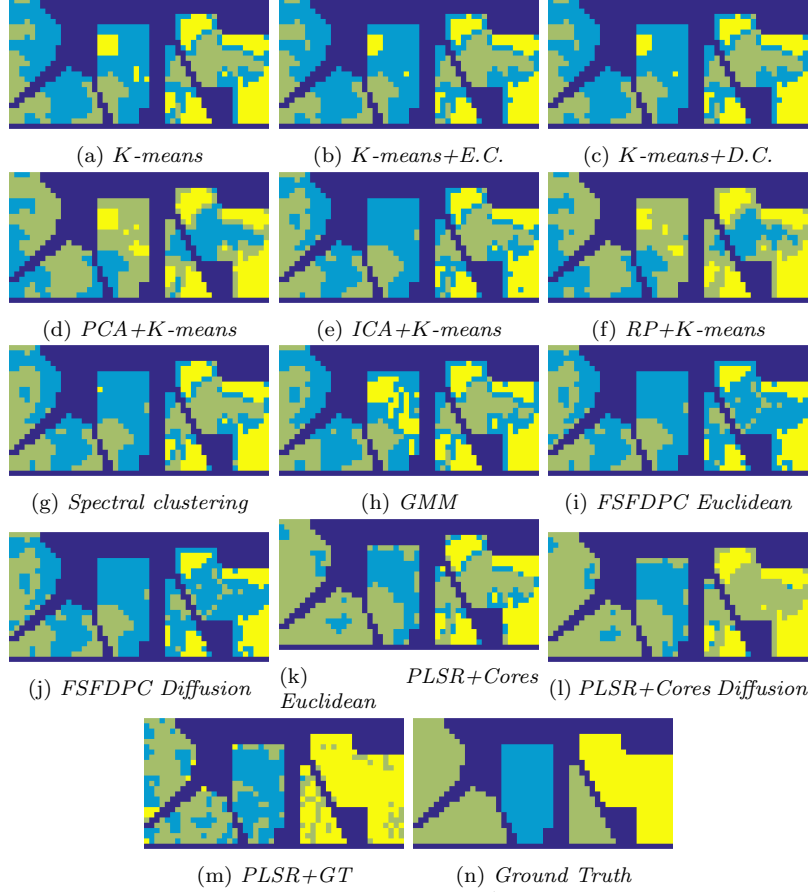
The Indian Pines dataset consists of three classes that are difficult to distinguish visually; see Figure 13. It is expected that this dataset will be challenging for all algorithms, due to the lack of clear separation between the classes. Results for Indian Pines appear in Figure 14.



**Figure 13:** The Indian Pines data is a  $50 \times 25$  subset of the full Indian Pines dataset. It contains 3 classes, one of which is not well-localized spatially. The dataset was captured in 1992 in Northwest IN, USA by the AVIRIS sensor. The spatial resolution is 20m/pixel. There are 200 spectral bands.

For the Indian Pines dataset, all methods have difficulty distinguishing between the clusters. The proposed method, PLSR with diffusion cores, provides the best quantitative results, followed by PLSR with Euclidean cores. Performance is not close to the performance of supervised PLSR, indicating that this is a challenging example. The learned cores are compared to the randomly generated training data in Figure 15. In this example, the three learned cores are not completely pure, despite the algorithm correctly identifying three modes from three different classes. Indeed, one can see in Figure

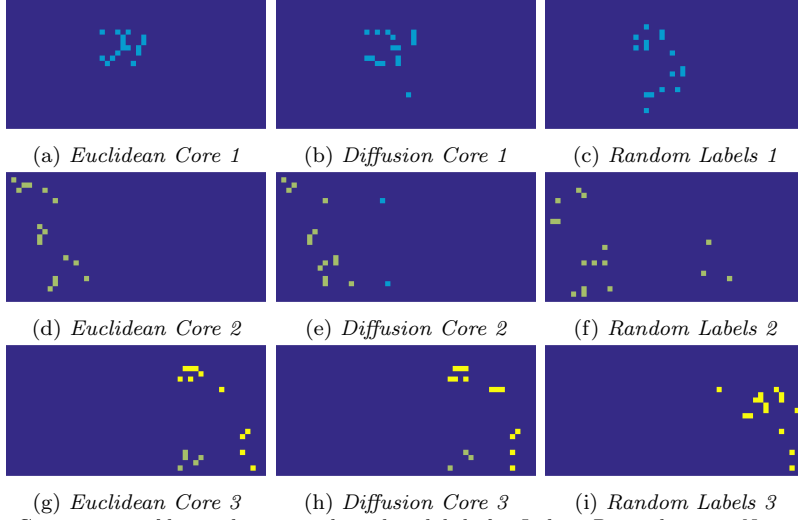
15 that learned core 3 is not totally pure. This indicates overlap of the spectra of class 2 and class 3, making this a challenging dataset to cluster with high accuracy. However, the majority of points from learned core 3 have ground truth label 3.



**Figure 14:** Clustering results for Indian Pines dataset. All methods have trouble with this example, particularly with the right portion of the ground truth. Interestingly, the supervised method PLSR with GT inaccurately labels the green portion of this part of the ground truth, while PLSR with the learned cores inaccurately labels the yellow part. Generally, PLSR with learned cores outperforms other methods by strongly improving labeling accuracy on the left part of the ground truth.

Metric	<i>K</i> -means	<i>K</i> -means+E.C.	<i>K</i> -means+D.C.	PCA+ <i>K</i> -means	ICA+ <i>K</i> -means	RP+ <i>K</i> -means	SC	GMM	FSFDPC-E.	FSFDPC-D.	PLSR+E.C.	PLSR+D.C.	PLSR+GT
OA	0.4465	0.4542	0.4542	0.4515	0.5218	0.4884	0.5246	0.5124	0.4794	0.4794	0.6401	<b>0.6521</b>	0.7305
AA	0.4530	0.4980	0.4980	0.4031	0.5504	0.4736	0.5548	0.5018	0.5140	0.5140	0.6121	<b>0.6197</b>	0.7493
$\kappa$	0.1703	0.2073	0.2073	0.1392	0.2886	0.2086	0.2924	0.2507	0.2546	0.2546	0.4237	<b>0.4278</b>	0.5917

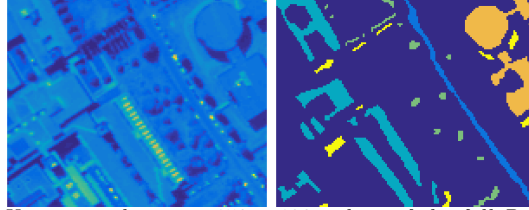
**Table 4:** Quantitative analysis for Indian Pines dataset.



**Figure 15:** Comparison of learned cores and random labels for Indian Pines dataset. Note that diffusion core 2 is impure, as is core 3 for both Euclidean and diffusion distances. This suggests there is challenging overlap with this example.

### 3.7. Pavia University Dataset

The Pavia University dataset contains five classes of wide-ranging size, many of which are spread throughout the image spatially. Figure 16 shows that the yellow, teal, and green classes are relatively small and quite well distributed throughout the image.

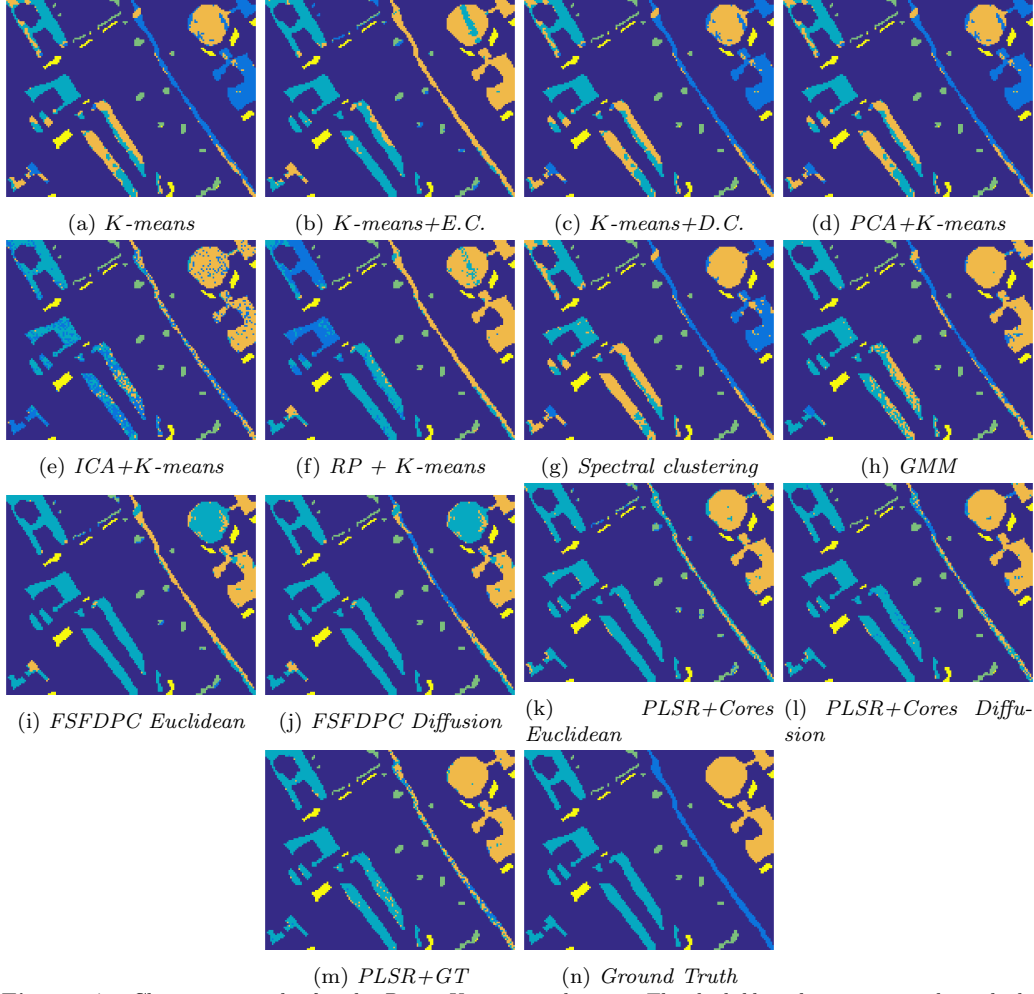


**Figure 16:** The Pavia University data is a  $150 \times 120$  subset of the full Pavia University dataset. It contains 5 classes, most of which are not well-localized spatially. The dataset was captured by the ROSIS sensor during a flight over Pavia, Italy. The spatial resolution is 1.3 m/pixel. There are 103 spectral bands.

For the Pavia University dataset, PLSR with diffusion cores performs best among unsupervised methods, followed by PLSR with Euclidean cores. In fact, the performance of PLSR with diffusion cores is comparable with that of supervised PLSR with GT in this example; the learned cores are quite similar to the uniformly sampled training data, as seen in Figure 18. The use of diffusion distance is advantageous over Euclidean distance in this example, and PLSR is advantageous over FSFDPC.

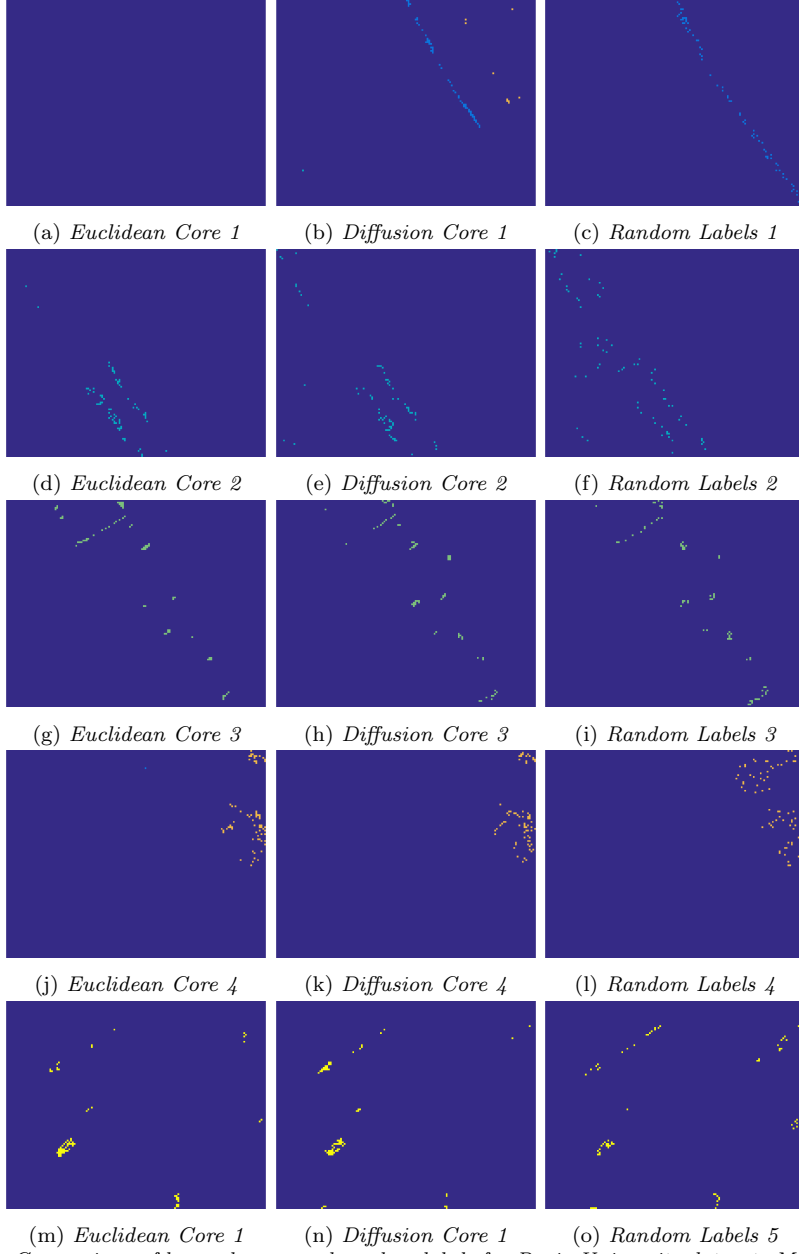
Metric	<i>K</i> -means	<i>K</i> -means+E.C.	<i>K</i> -means+D.C.	PCA+ <i>K</i> -means	ICA+ <i>K</i> -means	RP+ <i>K</i> -means	SC	GMM	FSFDPC-E.	FSFDPC-D.	PLSR+E.C.	PLSR+D.C.	PLSR+GT
OA	0.6096	0.7785	0.6102	0.6099	0.6749	0.6500	0.5926	0.7207	0.7255	0.7693	0.8297	<b>0.8650</b>	0.8607
AA	0.7471	0.6812	0.7474	0.7473	0.7567	0.7567	0.7437	0.7304	0.6663	0.7643	0.7206	<b>0.8158</b>	0.8038
$\kappa$	0.4728	0.6657	0.4735	0.4732	0.5588	0.5229	0.4503	0.5990	0.5693	0.6425	0.7365	<b>0.7976</b>	0.7895

**Table 5:** Quantitative analysis for Pavia University dataset.



**Figure 17:** Clustering results for the Pavia University dataset. The dark blue class running through the middle of the image and the two large teal regions near the bottom of the image appear to cause challenges for most methods, except for the FSFDPC with diffusion cores and the PLSR methods. However, FSFDPC with diffusion cores mislabels the circular region in the brown class, dropping its accuracy.

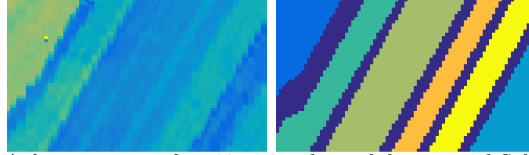




**Figure 18:** Comparison of learned cores and random labels for Pavia University dataset. Mode detection with Euclidean distances is unable to learn a core from class 1, and diffusion core 1 is impure. This indicates this is a difficult class to learn, which is affirmed by the clustering results in Figure 17, where this class is often mislabeled.

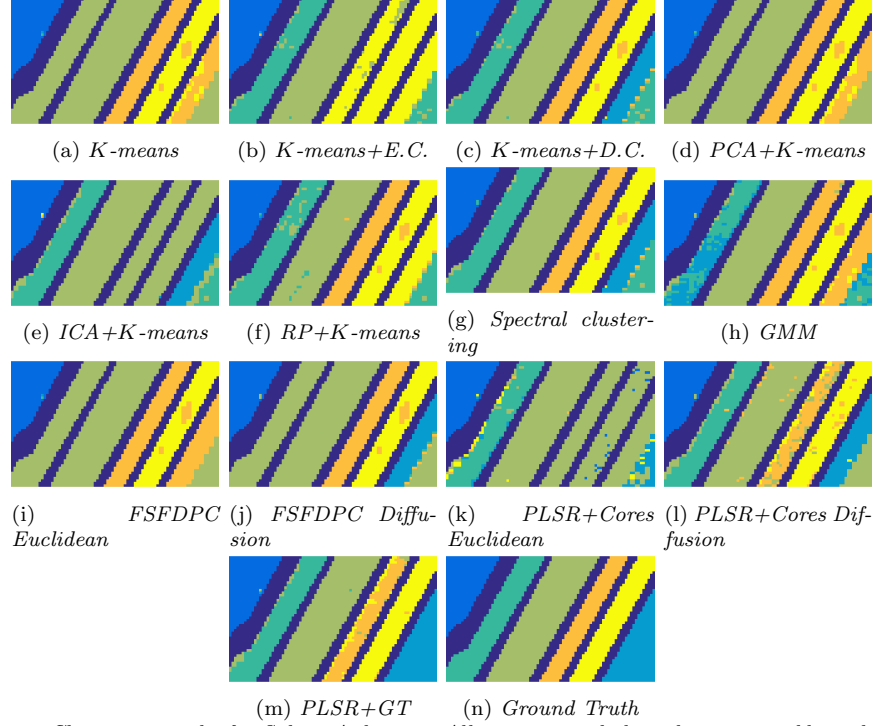
### 3.8. SalinasA Dataset

The Salinas A dataset consists in 6 classes arrayed in spatial rows. The dataset appears in Figure 19.



**Figure 19:** The Salinas A data consists of an  $80 \times 50$  subset of the original Salinas A dataset. It contains 6 classes, all of which are well-localized spatially. The dataset was captured over Salinas Valley, CA, by the AVIRIS sensor. The spatial resolution is 3.7 m/pixel. The image contains 224 spectral bands.

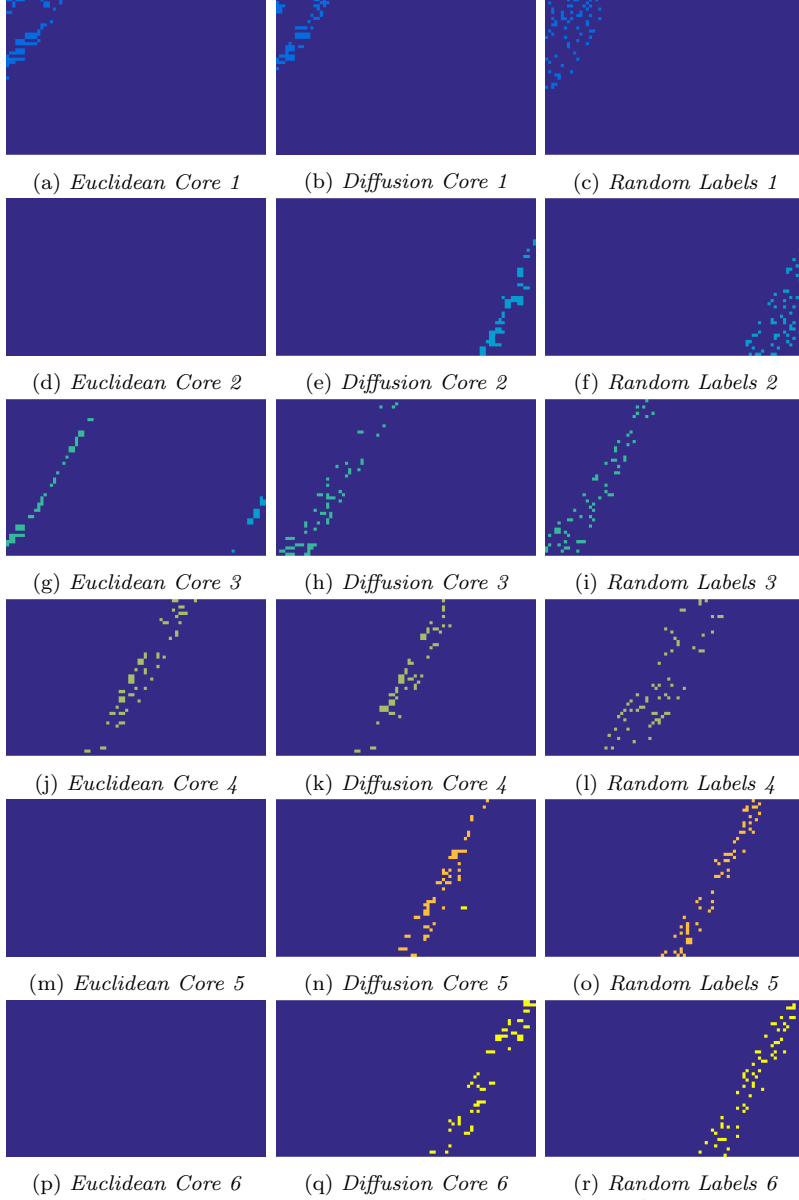
In the SalinasA dataset, spectral clustering performs best of the unsupervised methods, followed by  $K$ -means with diffusion cores. This example emphasizes the ways in which the proposed method can fail: even if the modes of the classes are learned correctly, the corresponding cores may not be sufficiently variable to cover the full diversity of the class. Indeed, in Figure 21, cores 1,2, and 3 are highly localized compared to the uniformly sampled training points, especially core 2. This leads to PLSR with diffusion cores being outperformed in this example.



**Figure 20:** Clustering results for SalinasA dataset. All unsupervised algorithms are unable to distinguish the class on the top right of the ground truth image, splitting it into two regions. The proposed algorithm correctly learns a unique core of points for this class, but that core was inadequate for capturing the full variance of the class. The supervised method, which randomly sampled the class to generate training data, correctly labeled the region. This class has high variance and multiple high density regions, which presents a challenge for the proposed unsupervised algorithm.

Metric	K-means	K-means+E.C.	K-means+D.C.	PCA+K-means	ICA+K-means	RP+K-means	SC	GMM	FSFDPC-E.	FSFDPC-D.	PLSR+E.C.	PLSR+D.C.	PLSR+GT
OA	0.6814	0.7325	0.9292	0.6818	0.6681	0.6855	<b>0.9327</b>	0.7768	0.6799	0.7611	0.5928	0.8711	0.9540
AA	0.6551	0.6570	0.9152	0.6556	0.5946	0.6545	<b>0.9190</b>	0.7534	0.6535	0.7558	0.5093	0.8485	0.9455
$\kappa$	0.5946	0.6669	0.9125	0.5952	0.5686	0.6001	<b>0.9169</b>	0.7242	0.5933	0.6962	0.4619	0.8390	0.9432

**Table 6**

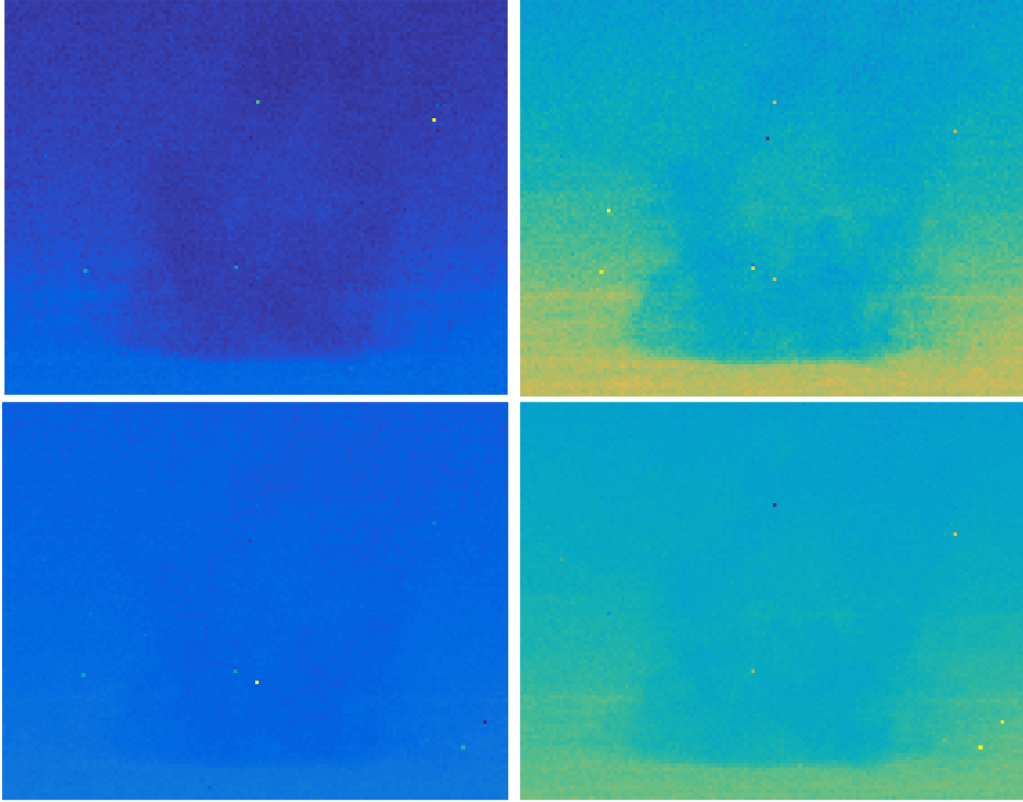


**Figure 21:** Comparison of learned cores and random labels for Salinas A dataset. Notice that cores 1, 3 and especially 2 are highly localized in space. This suggests that they may be insufficient to serve as a substitute for uniformly sampled training points, which would account for more of the within class variance.

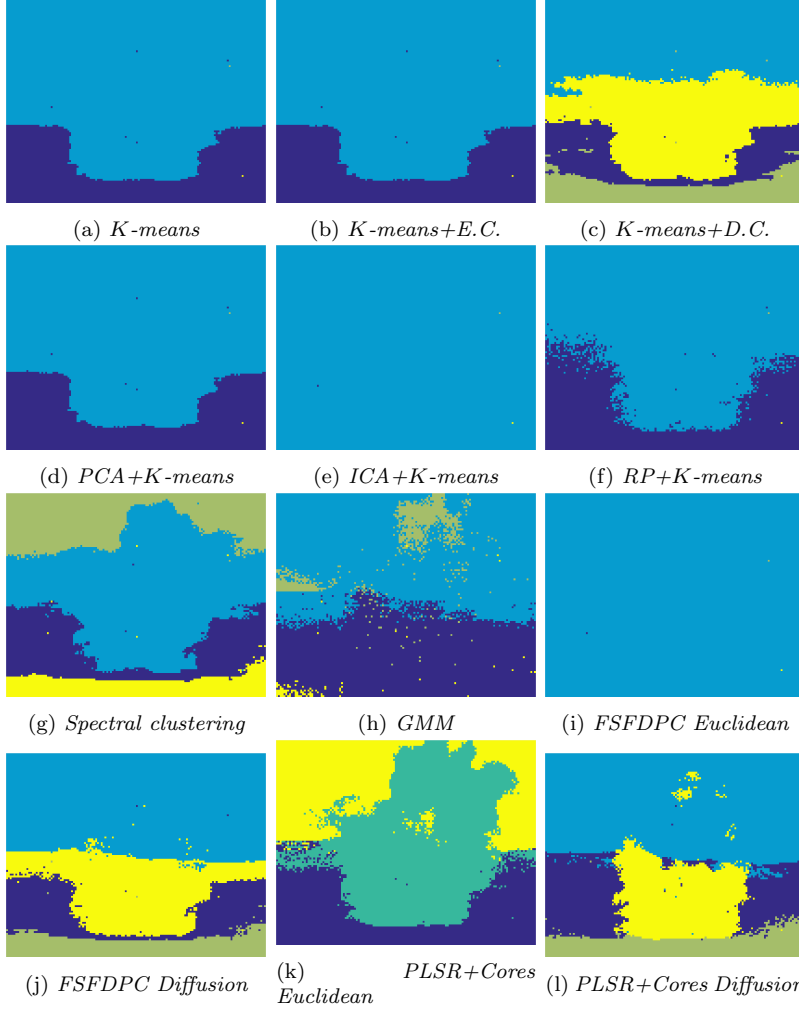
### 3.9. Chemical Plume Segmentation

As a real HSI final example, we consider the problem of chemical plume segmentation in HSI. Chemicals invisible to the naked eye can be captured

in certain frequency bands of an HSI; the problem is to identify the chemical region. An example, courtesy of the Johns Hopkins Applied Physics Lab, appears in Figure 22. Several pixels are corrupted, making this problem even more difficult. No labeled ground truth is available for this image, so we evaluate with only visual quality of the image segmentation. In order to determine the number of clusters,  $K$ , used by our clustering algorithms, we examine the plot of sorted  $\mathcal{D}(x_n) = p(x_n)\rho(x_n)$  values. We look for “kinks” in this plot, which correspond to a major drop in  $D(x_n)$  values. This heuristically indicates that, to a certain granularity, all clusters have been located. The plots of sorted  $\mathcal{D}(x_n)$  and the finite differences between the successive values appears in Figure 24. We notice two local extrema: one at  $n = 2$  and one at  $n = 4$ . The extrema at  $n = 2$  was hypothesized to separate the tops and bottoms of the image, so was not used for the purpose of chemical plume segmentation. Hence,  $K = 4$  was used as the estimated number of clusters to achieve chemical plume segmentation.

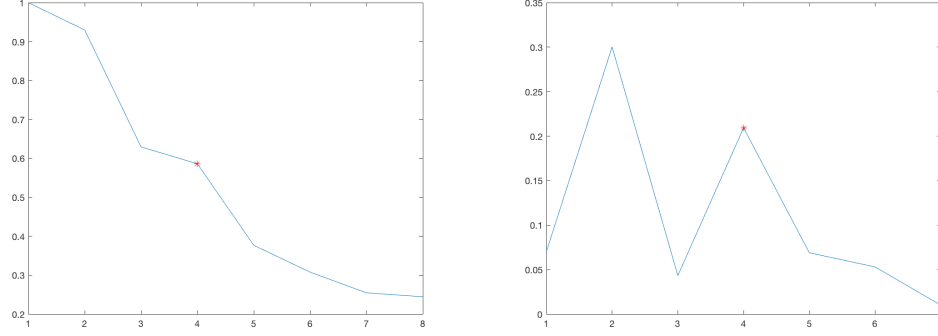


**Figure 22:** A chemical plume HSI dataset of spatial dimensions  $110 \times 140$ . There are 129 spectral bands. Shown, clockwise from top left: band 11, 16, 18, and 29. These bands were selected to demonstrate the visibility of the chemical plume in certain bands, but not in others.



**Figure 23:** Results for APL chemical plume dataset. Only the PLSR methods are able to achieve reasonable plume segmentation. PLSR with Euclidean distance, however, is unable to correctly segment horizontally, resulting in a plume that is spread too far. However, it correctly ascertains that the plume diffuses somewhat far in the vertical direction. The PLSR with diffusion cores correctly segments in the horizontal direction, but is too conservative vertically. Overall, PLSR with Euclidean cores is too expansive in its segmentation, while PLSR with diffusion cores is arguably too conservative. Several of the other the methods were unable to detect anomalous pixels effectively, resulting in clusters of very small sizes.

We see from Figure 23 that PLSR with diffusion cores segments the plume best, in particular giving the cleanest horizontal segmentation. PLSR with Euclidean cores performs second best, giving a good vertical segmentation, but making some confusion in the horizontal direction. Most of the other methods give very poor segmentation.



**Figure 24:** Plot of sorted  $\mathcal{D}(x_n)$  values for APL chemical plume HSI and a plot of the finite differences between successive values. The extrema of the difference curve were used to estimate the number of clusters  $K$  to use for plume segmentation. The value  $K = 4$  was used after analyzing these plots.

#### 4. Overall comments on the Experiments and Conclusion

The quantitative analyses of clustering real HSI experiments with ground truth are summarized in Table 7.

In general, using diffusion distances for mode detection offers improvement over using Euclidean distances. Indeed, this improvement can be substantial, as in the Pavia dataset when PLSR with diffusion cores achieves an overall accuracy of 0.9256, compared to just 0.6957 for PLSR with Euclidean cores. Moreover, the use of the learned cores for PLSR is generally more effective than using the learned modes in FSFDPC. Although FSFDPC with diffusion modes outperforms PLSR with diffusion cores for the KSC dataset, it does so by a fine margin in a case when both methods have overall accuracy exceeding 0.975. More significant differences in the methods occur for the Pavia, Indian Pines, and Pavia University datasets, where the improvements in overall accuracy from using PLSR with diffusion cores compared to FSFDPC with diffusion modes are 0.7169 to 0.9256, 0.4794 to 0.6521, and 0.7693 to 0.8650, respectively. Moreover, the improvement from using PLSR

Data	Metric	$K$ -means	$K$ -means+E.C.	$K$ -means+D.C.	PCA+ $K$ -means	ICA+ $K$ -means	RP+ $K$ -means	SC	GMM	FSFDPC-E	FSFDPC-D	PLSR+E.C.	PLSR+D.C.	PLSR+GT
KSC	OA	0.9448	0.9448	0.9448	0.9448	0.8267	0.8899	0.9532	0.8825	0.8390	<b>0.9831</b>	0.9755	0.9770	0.9805
KSC	AA	0.9320	0.9320	0.9320	0.9320	0.7377	0.8753	0.9427	0.8662	0.7422	<b>0.9754</b>	0.9684	0.9709	0.9743
KSC	$\kappa$	0.9242	0.9242	0.9242	0.9242	0.7584	0.8501	0.9359	0.8408	0.7751	<b>0.9768</b>	0.9663	0.9684	0.9733
Pavia	OA	0.6502	0.6502	0.8896	0.6502	0.8437	0.6434	0.7942	0.7118	0.7010	0.7169	0.6957	<b>0.9256</b>	0.9592
Pavia	AA	0.5356	0.5356	0.7731	0.5356	0.8754	0.5444	0.7531	0.6175	0.5688	0.7776	0.7052	<b>0.9504</b>	0.9659
Pavia	$\kappa$	0.5715	0.5725	0.8597	0.5717	0.8072	0.5642	0.7503	0.6435	0.6279	0.6471	0.6259	<b>0.9068</b>	0.9483
IP	OA	0.4465	0.4542	0.4542	0.4515	0.5218	0.4884	0.5246	0.5124	0.4794	0.4794	0.6401	<b>0.6521</b>	0.7305
IP	AA	0.4530	0.4980	0.4980	0.4031	0.5504	0.4736	0.5548	0.5018	0.5140	0.5140	0.6121	<b>0.6197</b>	0.7493
IP	$\kappa$	0.1703	0.2073	0.2073	0.1392	0.2886	0.2086	0.2924	0.2507	0.2546	0.2546	0.4237	<b>0.4278</b>	0.5917
PaviaU	OA	0.6096	0.7785	0.6102	0.6099	0.6749	0.6500	0.5926	0.7207	0.7255	0.7693	0.8297	<b>0.8650</b>	0.8607
PaviaU	AA	0.7471	0.6812	0.7474	0.7473	0.7567	0.7567	0.7437	0.7304	0.6663	0.7643	0.7206	<b>0.8158</b>	0.8038
PaviaU	$\kappa$	0.4728	0.6657	0.4735	0.4732	0.5588	0.5229	0.4503	0.5990	0.5993	0.6425	0.7365	<b>0.7976</b>	0.7895
SalinasA	OA	0.6814	0.7325	0.6292	0.6818	0.6681	0.6855	<b>0.9327</b>	0.7768	0.6799	0.7611	0.5928	0.8711	0.9540
SalinasA	AA	0.6551	0.6570	0.9152	0.6556	0.5946	0.6545	<b>0.9190</b>	0.7534	0.6335	0.7558	0.5093	0.8485	0.9455
SalinasA	$\kappa$	0.5946	0.6669	0.9125	0.5952	0.5686	0.6001	<b>0.9169</b>	0.7242	0.5933	0.6962	0.4619	0.8390	0.9432

**Table 7:** Summary of quantitative analyses of real HSI clustering. Generally the diffusion methods offer the strongest overall performance, particularly PLSR + diffusion cores. The notable exception is the SalinasA data, where the classes were of too high variance for the proposed methods to produce cores that could substitute for randomly sample training data.



instead of FSFDPC is not restricted to the case of diffusion modes and cores. Similar improvements occur in the case of modes and cores learned from Euclidean distance. The quantitative improvements extend to overall accuracy and  $\kappa$ .

In many of the visualizations of the clustering results, it is seen that PLSR with both Euclidean and diffusion distances tend to make errors near the boundaries of classes, when they err. This can be seen, for example, with the blue and green classes in the KSC dataset and with the brown class in the Pavia University dataset. Assuming that spatial boundaries tend to correspond with spectral boundaries, this makes sense: the mode estimation algorithm is likely to learn points far from the boundary of a class. This property of the proposed algorithm is another manifestation of an aforementioned drawback: the learned training points do not always cover the entire class, as randomly sampled training data often would. This may leave the tails of the class distributions unexplored, to the detriment of clustering.

The proposed methods do, however, have drawbacks. As can be seen from Figure 6, when the mode estimation algorithm fails to find the correct modes of the constituent distributions, the method fails dramatically. Moreover, the SalinasA experiments indicate that if a class has large variance, then learning a mode and taking its nearest neighbors may be an insufficient substitute for a uniformly sampled random training set. This is because important parts of the distribution may be missed, which due to the large within class variance, may be quite unlike the learned core.

#### 4.1. Computational Complexity

The proposed method consists of two major steps. First, the modes of the data must be estimated, then PLSR must be performed. Suppose the data  $X$  is  $N \times D$ , with  $D \ll N$ . To compute the modes of the data, the most expensive part of the process is the construction of the diffusion distance matrix. The worst-case computational complexity of diffusion maps is  $\tilde{O}(C_d K D N)$ , where  $K$  is the number of eigenvectors computed, and  $C_d$  is a constant that depends exponentially in the intrinsic dimension of the data, by using fast nearest neighbor algorithms such as covertrees [45], and  $\tilde{O}$  hides logarithmic factors. For all examples presented in this paper,  $K \ll N$ , and can in general be assumed  $O(1)$  independent of  $N, D$ .

The PLSR is implemented with the SIMPLS algorithm [29], which has computational complexity  $O(DNm + Dm^2 + m^3)$ , where  $m$  is the number of partial least squares components used. In our algorithm, this is equal to

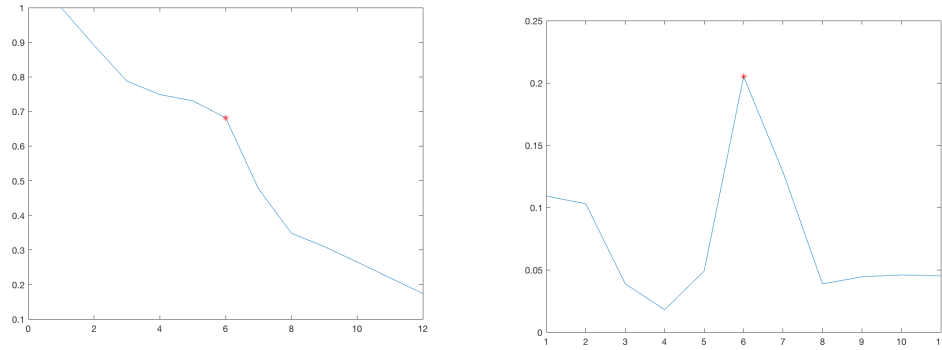
the number of classes, either known or determined by studying the decay of the sorted  $\mathcal{D}(x_n)$  curve, and may be assumed to be  $O(1)$  and independent of  $N, D$ . Thus, the overall cost of our algorithm is  $\tilde{O}(C_d DN)$ , i.e. essentially optimal in the input data size  $DN$ , as soon as the intrinsic dimension  $d$  of the data is small.

This is comparable to spectral clustering, which has the same eigenvector bottleneck as diffusion maps, and to the FSFDPC algorithm, which in general requires  $O(C_d DN)$  operation to compute each point’s distance to its nearest neighbor of higher density.

Both the methods are described in detail in the Appendix.

## 5. Future Research Directions

A drawback of many clustering algorithms, including the one presented in this paper, is the assumption that the number of clusters,  $K$ , is known a priori. Initial investigations suggest that looking for the “kink” in the sorted plot of  $\mathcal{D}(x_n)$  could be used to detect  $K$  automatically. Indeed, this is true for most of the examples presented in this article. See Figure 25 for an example from the Pavia dataset.



(a) Plot of the largest 10 values of  $\mathcal{D}(x_n)$ , sorted from largest to smallest. It may be possible in some cases to use the largest drop-off as an estimate of the number of clusters,  $K$ . The largest drop-off is indicated with a red star. (b) The difference between successive values of sorted values of  $\mathcal{D}(x_n)$ . The largest drop correspond to the correct number of classes, as determined by the ground truth and labeled with a red star.

**Figure 25**

This was suggested in [23] in the case of using Euclidean distances to compute  $\rho$ , however this seems to impose unrealistic constraints on the relative positions of the mixture components. It is of interest to prove under what assumptions on the distributions and mixture model the plot  $\mathcal{D}(x_n)$  correctly determines  $K$ . Moreover, in the case that one cluster is noticeably smaller

or harder to detect than others, as in the case of the APL chemical plume dataset, it may be advantageous to use a different local maximum of the finite difference curve, rather than the global maximum.

Another drawback of the proposed method, illustrated clearly in 3.8, is that the learned core may not be a serviceable substitute for uniformly sampled training data if the class being learned is of sufficiently high variance. The learned cores are necessarily low variance, since they are the nearest neighbors of a high density point. Finding a way to get a few points further along the tails, in addition to the learned core, could significantly boost the proposed algorithm for data with large within class variance.

Relatedly, the present work is essentially heuristic; it is not known mathematically under what constraints on the mixture model the method proposed for learning modes succeeds with high probability. Besides being of mathematical interest, this would be useful for understanding the limitations of the proposed method for HSI. Indeed, the experimental results of this paper indicate that diffusion distance is often more effective than Euclidean distance for discriminating between high density points for use in the FSFDPC algorithm. To understand this phenomenon rigorously, a careful analysis of diffusion distances for data drawn from a non-parametric mixture model is required. This is the subject of ongoing mathematical research.

### *5.1. Acknowledgments*

This research was partially supported by NSF-ATD-1222567, AFOSR FA9550-14-1-0033, and NSF-IIS-1546392.

## References

- [1] Lu, G.; Fei, B. Medical hyperspectral imaging: a review. *Journal of biomedical optics* **2014**, *19*, 010901–010901.
- [2] Wang, Y.; Chen, G.; Maggioni, M. High-Dimensional Data Modeling Techniques for Detection of Chemical Plumes and Anomalies in Hyperspectral Images and Movies. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **2016**, *9*, 4316–4324.
- [3] Eismann, M. Hyperspectral remote sensing. SPIE Bellingham, 2012.
- [4] Ma, L.; Crawford, M.; Tian, J. Local manifold learning-based  $k$ -nearest-neighbor for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing* **2010**, *48*, 4099–4109.
- [5] Li, W.; Tramel, E.; Prasad, S.; Fowler, J. Nearest regularized subspace for hyperspectral classification. *IEEE Transactions on Geoscience and Remote Sensing* **2014**, *52*, 477–489.
- [6] Melgani, F.; Bruzzone, L. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on Geoscience and Remote Sensing* **2004**, *42*, 1778–1790.
- [7] Fauvel, M.; Benediktsson, J.; Chanussot, J.; Sveinsson, J. Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles. *IEEE Transactions on Geoscience and Remote Sensing* **2008**, *46*, 3804–3814.
- [8] Ratle, F.; Camps-Valls, G.; Weston, J. Semisupervised neural networks for efficient hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing* **2010**, *48*, 2271–2282.
- [9] Chen, Y.; Jiang, H.; Li, C.; Jia, X.; Ghamisi, P. Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing* **2016**, *54*, 6232–6251.
- [10] Liang, H.; Li, Q. Hyperspectral imagery classification using sparse representations of convolutional neural network features. *Remote Sensing* **2016**, *8*, 99.

- [11] Qian, Y.; Ye, M.; Zhou, J. Hyperspectral image classification based on structured sparse logistic regression and three-dimensional wavelet texture features. *IEEE Transactions on Geoscience and Remote Sensing* **2013**, *51*, 2276–2291.
- [12] Li, J.; Bioucas-Dias, J.; Plaza, A. Semisupervised hyperspectral image classification using soft sparse multinomial logistic regression. *IEEE Geoscience and Remote Sensing Letters* **2013**, *10*, 318–322.
- [13] Bioucas-Dias, J.; Plaza, A.; Dobigeon, N.; Parente, M.; Du, Q.; Gader, P.; Chanussot, J. Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **2012**, *5*, 354–379.
- [14] Jia, S.; Qian, Y. Constrained nonnegative matrix factorization for hyperspectral unmixing. *IEEE Transactions on Geoscience and Remote Sensing* **2009**, *47*, 161–173.
- [15] Févotte, C.N.D. Nonlinear hyperspectral unmixing with robust non-negative matrix factorization. *IEEE Transactions on Image Processing* **2015**, *24*, 4810–4819.
- [16] Bioucas-Dias, J.; Figueiredo, M. Alternating direction algorithms for constrained sparse regression: Application to hyperspectral unmixing. 2nd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS). IEEE, 2010, pp. 1–4.
- [17] Li, C.; Ma, Y.; Mei, X.; Liu, C.; Ma, J. Hyperspectral unmixing with robust collaborative sparse regression. *Remote Sensing* **2016**, *8*, 588.
- [18] Gillis, N.; Kuang, D.; Park, H. Hierarchical clustering of hyperspectral images using rank-two nonnegative matrix factorization. *IEEE Transactions on Geoscience and Remote Sensing* **2015**, *53*, 2066–2078.
- [19] Paoli, A.; Melgani, F.; Pasolli, E. Clustering of hyperspectral images based on multiobjective particle swarm optimization. *IEEE transactions on geoscience and remote sensing* **2009**, *47*, 4175–4188.

- [20] Chen, Y.; Ma, S.; Chen, X.; Ghamisi, P. Hyperspectral data clustering based on density analysis ensemble. *Remote Sensing Letters* **2017**, *8*, 194–203.
- [21] Acito, N.; Corsini, G.; Diani, M. An unsupervised algorithm for hyperspectral image segmentation based on the Gaussian mixture model. IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2003, Vol. 6, pp. 3745–3747.
- [22] Cariou, C.; Chehdi, K. Unsupervised nearest neighbors clustering with application to hyperspectral images. *IEEE Journal of Selected Topics in Signal Processing* **2015**, *9*, 1105–1116.
- [23] Rodriguez, A.; Laio, A. Clustering by fast search and find of density peaks. *Science* **2014**, *344*, 1492–1496.
- [24] Zhang, H.; Zhai, H.; Li, L.Z.P. Spectral–spatial sparse subspace clustering for hyperspectral remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing* **2016**, *54*, 3672–3684.
- [25] R.R., C.; Lafon, S.; Lee, A.; Maggioni, M.; Nadler, B.; Warner, F.; Zucker, S. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences of the United States of America* **2005**, *102*, 7426–7431.
- [26] Coifman, R.; Lafon, S. Diffusion maps. *Applied and computational harmonic analysis* **2006**, *21*, 5–30.
- [27] Wold, H. Partial least squares. *Encyclopedia of statistical sciences* **1985**.
- [28] Geladi, P.; Kowalski, B. Partial least-squares regression: a tutorial. *Analytica chimica acta* **1986**, *185*, 1–17.
- [29] De Jong, S. SIMPLS: an alternative approach to partial least squares regression. *Chemometrics and intelligent laboratory systems* **1993**, *18*, 251–263.
- [30] Friedman, J.; Hastie, T.; Tibshirani, R. *The Elements of Statistical Learning*; Vol. 1, Springer series in statistics Springer, Berlin, 2001.

- [31] Wasserman, L. *All of statistics: a concise course in statistical inference*; Springer Science & Business Media, 2013.
- [32] Hansen, P.; Schjoerring, J. Reflectance measurement of canopy biomass and nitrogen status in wheat crops using normalized difference vegetation indices and partial least squares regression. *Remote sensing of environment* **2003**, *86*, 542–553.
- [33] Li, X.; Zhang, Y.; Bao, Y.; Luo, J.; Jin, X.; Xu, X.; Song, X.; Yang, G. Exploring the best hyperspectral features for LAI estimation using partial least squares regression. *Remote Sensing* **2014**, *6*, 6221–6241.
- [34] Barker, M.; Rayens, W. Partial least squares for discrimination. *Journal of chemometrics* **2003**, *17*, 166–173.
- [35] Banerjee, M.; Capozzoli, M.; McSweeney, L.; Sinha, D. Beyond kappa: A review of interrater agreement measures. *Canadian journal of statistics* **1999**, *27*, 3–23.
- [36] Comon, P. Independent component analysis, a new concept? *Signal processing* **1994**, *36*, 287–314.
- [37] Hyvärinen, A.; Oja, E. Independent component analysis: algorithms and applications. *Neural networks* **2000**, *13*, 411–430.
- [38] Dasgupta, S. Experiments with random projection. Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann Publishers Inc., 2000, pp. 143–151.
- [39] Candes, E.; Tao, T. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE transactions on information theory* **2006**, *52*, 5406–5425.
- [40] Ng, A.; Jordan, M.; Weiss, Y. On spectral clustering: Analysis and an algorithm. NIPS, 2001, Vol. 14, pp. 849–856.
- [41] Von Luxburg, U. A tutorial on spectral clustering. *Statistics and computing* **2007**, *17*, 395–416.
- [42] Fauvel, M.; Tarabalka, Y.; Benediktsson, J.; Chanussot, J.; Tilton, J. Advances in spectral-spatial classification of hyperspectral images. *Proceedings of the IEEE* **2013**, *101*, 652–675.

- [43] Manolakis, D.; Siracusa, C.; Shaw, G. Hyperspectral subpixel target detection using the linear mixing model. *IEEE Transactions on Geoscience and Remote Sensing* **2001**, *39*, 1392–1409.
- [44] Kraut, S.; Scharf, L.; McWhorter, L. Adaptive subspace detectors. *IEEE Transactions on signal processing* **2001**, *49*, 1–16.
- [45] Beygelzimer, A.; Kakade, S.; Langford, J. Cover trees for nearest neighbor. International Conference on Machine Learning, 2006, pp. 97–104.
- [46] Rohrdanz, M.; Zheng, W.; Maggioni, M.; Clementi, C. Determination of reaction coordinates via locally scaled diffusion map. *The Journal of chemical physics* **2011**, *134*, 03B624.

## 6. Appendix

### 6.1. Diffusion Distance

We present a detailed overview of the diffusion maps construction. Analysis and implementation appear in [25, 26, 46]. Let  $X = \{x_n\}_{n=1}^N \subset \mathbb{R}^D$ . The diffusion distance [25, 26] between  $x, y \in X$ , denoted  $d_t(x, y)$ , is a notion of distance that incorporates and is uniquely determined by the underlying geometry of  $X$ . The construction of  $d_t$  involves constructing a weighted, undirected graph  $\mathcal{G}$  that encodes the geometry of  $X$ . The vertices of  $\mathcal{G}$  correspond to the points  $\{x_n\}_{n=1}^N$ , and the edges are computed as follows. Define an  $N \times N$  weight matrix  $W$  as

$$W_{ij} = \begin{cases} e^{-\frac{\|x_i - x_j\|_2^2}{\sigma}}, & i \in NN(j) \\ 0, & \text{else} \end{cases}$$

for some suitable choice of  $\sigma$  and for  $NN(j)$  the set of  $k$ -nearest neighbors of  $x_j$  in  $X$  in the  $\ell^2$  metric. In general, the choices of  $\sigma, k$  may be important; for all experiments shown in Section 3,  $\sigma = 1, k = 100$ . The edge in  $\mathcal{G}$  connecting vertices  $i$  and  $j$  has weight  $W_{ij}$ . Note that  $W$  is not automatically symmetric, so  $W^T + W$  is considered; other symmetrizations are possible. The symmetric weight matrix is then re-normalized so as to be independent of the density of  $X$  [26]. This yields an  $N \times N$  kernel matrix  $\mathcal{K}$  that is like a discrete



approximation to the Laplace-Beltrami operator on a manifold on which the sample points lie.

One then computes a spectral decomposition of  $\mathcal{K}$  to acquire a set of eigenvalues  $1 = \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N \geq 0$  with corresponding eigenvectors  $\{\Phi_n\}_{n=1}^N$ . The *diffusion distance* is

$$d_t(x, y) = \sqrt{\sum_{i=1}^N |\mathcal{K}(x, x_i) - \mathcal{K}(y, x_i)|^2} = \sqrt{\sum_{n=1}^N \lambda_n^{2t} (\Phi_n(x) - \Phi_n(y))^2}.$$

Since  $\lambda_n \leq 1$ , for large  $t$  many terms of the sum are close to 0, and the sum may be approximated by its truncation at index  $M$ , for some suitable  $2 \leq M \leq N$ . In all experiments,  $M$  was set to be the value at which the decay of the eigenvalues  $\{\lambda_n\}_{n=1}^N$  began to taper off, i.e. the “kink” in the eigenvalue plot. This is a standard heuristic for such metrics. Intuitively, the larger eigenvalues contain significant information, while the smaller ones are mostly noise and are of little value. The diffusion distance is parametrized by  $t$ , which encodes how long the diffusion process on the graph has run when the distances are computed. Small values of  $t$  allow a small amount of diffusion, which prevents the interesting geometry of  $X$  from being discovered. Large values of  $t$  allow the diffusion process to run for so long that the interesting geometry is washed out. We settle on a compromise, and let  $t = 3$  for all numerical experiments.

One can think of the weighted eigenvectors  $\{\lambda_n^t \Phi_n\}_{n=1}^N$  as encoding new data-dependent, yet close to being geometrically intrinsic, coordinates of  $X$ , so that diffusion distance is simply the normal  $\ell^2$  distance, but in these new coordinates, rather than the usual Euclidean coordinates. The subset  $\{\lambda_n^t \Phi_n\}_{n=1}^M$  used in the computation of  $d_t$  is a dimension-reduced set of diffusion coordinates.

## 6.2. Partial Least Squares Regression

Partial least squares [27, 28, 29, 30] is a linear regression method that attempts to maximize variation in both the predictor and response variables; this is in contrast to principal component regression, which attempts to maximize variation in only the predictor variables. Assume some  $N \times D$  normalized and centered training data  $X = (X_1|X_2|\dots|X_D)$  is given, along with  $N \times K$  labels  $Y$ . PLS starts by computing the univariate regression

coefficient  $\hat{\phi}_{1,j}$  of  $Y$  on each column  $X_j$  of  $X$ . Summing, we acquire a first partial least squares direction

$$Z_1 = \sum_{j=1}^D \hat{\phi}_{1,j} X_j.$$

For the remaining PLS directions, the inputs  $X_j$  are weighted according to the strength of the univariate impact on  $Y$ . The outcome is regressed on  $Z_1$ , to obtain coefficient  $\hat{\theta}_1$ . The columns of  $X$  are then orthogonalized with respect to  $Z_1$ , and we repeat. We stop when a sufficient number of directions  $M \leq D$  are obtained, yielding directions  $Z_1, \dots, Z_M$ . In our experiments, we take  $M = K_{GT}$  when  $K_{GT}$  is known, or  $M$  is estimated by the kink in the  $\mathcal{D}(x_n)$  curve as above.

It is helpful to compare PLS to PCA. Let  $\Sigma = X^T X$  be the sample covariance matrix of  $X$ . In PCA, a representation of  $X$  is computed in which the  $m^{th}$  principle component is given by

$$v_m = \arg \max_{\substack{\|\alpha\|=1, \\ v_\ell^T \Sigma \alpha = 0, \\ \ell=0,1,\dots,m-1}} \text{Var}(X\alpha),$$

The conditions  $v_\ell^T \Sigma \alpha = 0$ ,  $\ell = 0, 1, \dots, m-1$  ensure  $Z_m = X\alpha$  is uncorrelated with all lower  $Z_\ell = Xv_\ell$ .

For partial least squares, the  $m^{th}$  PLS direction is given by

$$\phi_m = \arg \max_{\substack{\|\alpha\|=1, \\ \hat{\phi}_\ell^T \Sigma \alpha = 0, \\ \ell=0,1,\dots,m-1}} \text{Corr}^2(Y, X\alpha) \text{Var}(X\alpha),$$

Letting  $Z = (Z_1|Z_2|\dots|Z_M)$ , the PLSR coefficient  $\hat{\beta}$  satisfies the condition  $\hat{\beta}P^T = (Z^T Z)^{-1}Z^T Y$ , where  $P$  is the least-squares solution to overdetermined system  $ZP = X$ .